ION Script User's Guide

RSI
Research Systems Inc.

ION Version 6.1
July 2004 Edition
Copyright © Research Systems, Inc.
All Rights Reserved

0704ION61SCR

Restricted Rights Notice

The IDL®, ION Script™, and ION Java™ software programs and the accompanying procedures, functions, and documentation described herein are sold under license agreement. Their use, duplication, and disclosure are subject to the restrictions stated in the license agreement. Research Systems, Inc., reserves the right to make changes to this document at any time and without notice.

Limitation of Warranty

Research Systems, Inc. makes no warranties, either express or implied, as to any matter not expressly set forth in the license agreement, including without limitation the condition of the software, merchantability, or fitness for any particular purpose.

Research Systems, Inc. shall not be liable for any direct, consequential, or other damages suffered by the Licensee or any others resulting from use of the IDL or ION software packages or their documentation.

Permission to Reproduce this Manual

If you are a licensed user of this product, Research Systems, Inc. grants you a limited, nontransferable license to reproduce this particular document provided such copies are for your use only and are not sold or distributed to third parties. All such copies must contain the title page and this notice page in their entirety.

Acknowledgments

IDL® is a registered trademark and ION™, ION Script™, ION Java™, are trademarks of Research Systems Inc., registered in the United States Patent and Trademark Office, for the computer program described herein.

Numerical Recipes™ is a trademark of Numerical Recipes Software. Numerical Recipes routines are used by permission.

GRG2™ is a trademark of Windward Technologies, Inc. The GRG2 software for nonlinear optimization is used by permission.

NCSA Hierarchical Data Format (HDF) Software Library and Utilities
Copyright 1988-2001 The Board of Trustees of the University of Illinois
All rights reserved.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities
Copyright 1998-2002 by the Board of Trustees of the University of Illinois. All rights reserved.

CDF Library
Copyright © 2002 National Space Science Data Center
NASA/Goddard Space Flight Center

NetCDF Library
Copyright © 1993-1999 University Corporation for Atmospheric Research/Unidata

HDF EOS Library
Copyright © 1996 Hughes and Applied Research Corporation

This software is based in part on the work of the Independent JPEG Group.

Portions of this software are copyrighted by DataDirect Technologies, 1991-2003.

Portions of this software were developed using Unisearch's Kakadu software, for which Kodak has a commercial license. Kakadu Software. Copyright © 2001. The University of New South Wales, UNSW, Sydney NSW 2052, Australia, and Unisearch Ltd, Australia.

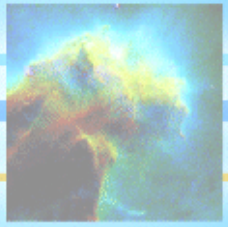
Portions of this computer program are copyright © 1995-1999 LizardTech, Inc. All rights reserved. MrSID is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Portions of this software are copyrighted by Merge Technologies Incorporated.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/)

IDL Wavelet Toolkit Copyright © 2002 Christopher Torrence.

Other trademarks and registered trademarks are the property of the respective trademark holders.



Contents

Chapter 1:
Configuring ION Script 9

Configuring ION Script for Windows 10
- The Images Tab 10
- The URLs Tab 11
- The Files Tab 12
- The Format Tab 13
- The Debug Tab 13

Configuring ION Script for UNIX 14
- Manually Configuring Your UNIX Web Server 16

ION Script Configuration Options 18
- Images 18
- URLs 19
- Files 20
- Format 23
- Debug 24

Chapter 2:
Overview 27
What Is ION Script? 28
ION Script Pages 28
ION Script Applications 32
Anatomy of an ION Script Application 33
Example 34
Requesting an ION Script Page 36
Where to Store Your Files 38
How HTML Files Are Normally Requested 38
ION Script Files 38
Keeping .ion and .html Files in the Same Place 39
IDL Files 39
Running the ION Script Examples 41

Chapter 3:
Variables, Expressions & Operators 43
Variables 44
Declaring & Assigning Values 44
Variable Types 44
Assigning String Literals to ION Script Variables 46
Determining Variable Type 46
Null Strings Versus Undefined Strings 46
Variable Names 47
Variable Scope 47
Variable Persistence 49
System Variables 50
$Browser 50
$Document 51
$Form 53
$FormURL 54
$ION 58
$Mouse 59
Expressions 61
Values 61
Expression Types 61

Operators 64
Mathematical Operators 64
String Operators 64
Comparison Operators 65
Logical Operators 66
Operator Precedence 67

Chapter 4:
Creating ION Script Applications 69

Specifying URLs 70
HTTP URLs 71
HTTPS URLs 71
File URLs 72
ION URLs 72
Absolute Paths vs. the ION Search Path 72
HTML Forms vs. ION Script Forms 74
Passing Name/Value Pairs in a URL 75
Passing ION Script Variables in a URL 75
Passing Multiple Values With the Same Name 76
Handling Multiple Selections in a SELECT Element 77
Parsing the $Form Variable in an <IDL> Block 77
Parsing the $Form Variable in a <SCRIPT> Block 78
Passing IDL Variables to ION Script 80
Using Frames with ION Script 82
Overview of Fixed Framesets 82
Loading ION Script Pages in Frames 84
Floating Frames 87
Targeting Different Frames 88
Accommodating All Browsers 89
Using JavaScript and VBScript 90
Graphics in ION Script 91
Direct Graphics 91
Object Graphics 91
Bandwidth Issues 92
Output Formats 92
Using Special and International Characters 94

Example: Creating a Complete Application 95
Step 1: Displaying an IDL-Generated Image 96
Step 2: Declaring and Displaying Variables 97
Step 3: Adding Interactivity 100
Step 4: Validating Form Data 104
Step 5: Creating Reusable Pages 106
Step 6: Creating Interactive Images for Data Picking 107
Step 7: Displaying IDL-Generated Data 108
A Note On Using the Back Button 110
Application Variations 110

Chapter 5:
ION Script Tag Reference 113

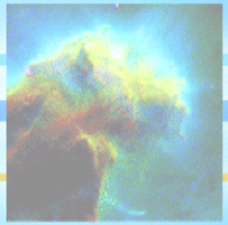
Syntax Conventions 114
Elements of Syntax 115
Case Sensitivity 116
Using Quotation Marks 116
Variable Substitution in Attribute Values 117
HTML Mappings 118
The HTML Comment Tag 119
Alphabetical Listing of ION Script Tags 121
EVENT_DECL 122
EVENTS 123
IDL 124
IDL Tag Limitations 124
Using ION_EVALUTE and ION_VARIABLE Tags in an IDL Block 125
Single vs. Double Quotation Marks 126
Commenting IDL Code 127
Using the $ Symbol in an IDL Block 127
INPUT 129
ION_BODY 133
ION_BUTTON 135
ION_CHECKBOX 138
ION_DATA_OUT 141
ION_EVALUATE 149
ION_FORM 152

Forms Containing a Single Text Field 153
ION_HEADER 156
Metadata Tags 156
Events and Variables 157
ION_IF, ION_ELSEIF, ION_ELSE 158
ION_IMAGE 159
Notes About Using ION_IMAGE 160
ION_INCLUDE 165
ION_LINK 170
ION_OBJECT 173
ION_PARAM 178
ION_RADIO 179
ION_SCRIPT 182
ION_VARIABLE 183
VARIABLE_DECL 186
VARIABLES 189

Chapter 6: Troubleshooting 191

Web Server Problems 192
If You Are Using Microsoft IIS 192
“Connection Failed” Errors 192
“Not Found” Errors 193
“No Event or Page” Errors 194
Licensing Errors 194
ION Script Syntax Errors 194
IDL Errors 195
Variables 196
Images Not Displaying 196
Frames 197

Index 199



Chapter 1:
Configuring ION Script

This chapter discusses the configuration tasks that must be completed before ION Script will run, as well as preferences that can be set to control the behavior of ION Script. The following topics are covered in this chapter:

- Configuring ION Script for Windows
- Configuring ION Script for UNIX
- ION Script Configuration Options

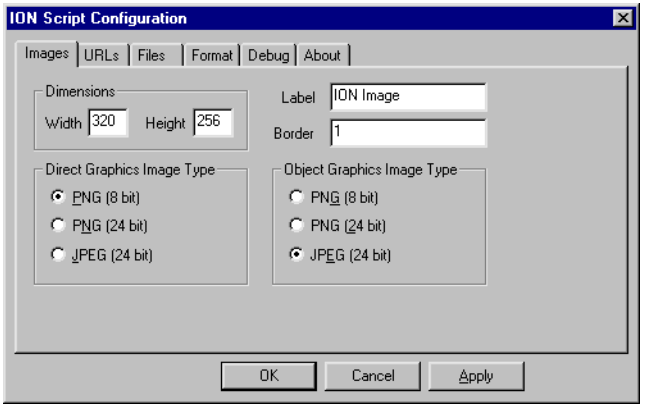
Configuring ION Script for Windows

The program `IONScriptConfig.exe` is used to specify various default settings, preferences, and file paths for the Windows version of ION Script. To configure ION Script for Windows, do the following:

1. Run `IONScriptConfig.exe`. You can access this program from the Windows Start menu under **Programs → Research Systems ION 6.1 → ION Script Configuration**. This program is located in the *`RSI-DIR`*`/IDL61/products/ION61/ion_script/bin` directory, where *`RSI-DIR`* is the directory in which you installed IDL.
2. Modify the desired settings on each of the tabs described below. Refer to "ION Script Configuration Options" on page 18 for a description of each configuration option.
3. When you have finished making changes, click **Apply**. The changes will take effect immediately; you do not need to restart Windows.

The Images Tab

The Images tab, shown below, is used to specify default properties for images created with the ION_IMAGE tag. Attributes of the ION_IMAGE tag can be used to override the default values specified on the Images tab on a case-by-case basis. For information on the attributes of ION_IMAGE, see "ION_IMAGE" on page 159.

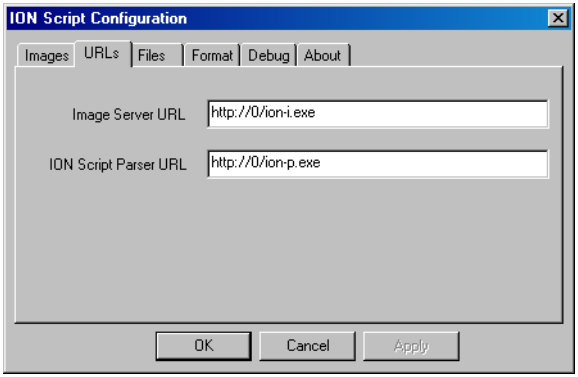


The URLs Tab

The URLs tab, shown below, is used to specify the path to the ION Script Image Server and the parser. The Image Server is part of the file `ion-i.exe`.

Note

During the installation process, you were prompted on the "Web Server Configuration" dialog for your Web server's fully qualified domain name and the proper path to your Web server's CGI executables directory. If you specified these values at that time, these values should already appear on the URLs tab, and you do not need to do anything further on the URLs tab.



If you did not specify this value on the Web Server Configuration dialog during the installation process, the Image Server field will be given a URL of the form:

```
http://WindowsMachineName/cgi-bin/ion-i.exe
```

where WindowsMachineName is the name of your Windows machine, and `/cgi-bin` is the path to the CGI executables directory for your Web server. Similarly, for the ION Script Parser URL field, you will see a URL of the form:

```
http://WindowsMachineName/cgi-bin/ion-p.exe
```

If the path to your Web server's CGI directory is different than `/cgi-bin`, change this to the correct path for all three fields.

Next, change the Windows machine name to the fully qualified domain name of your Web server. For example, if your hostname is myhost and your domain is mydomain.com, you would replace the Windows machine name with myhost.mydomain.com. The Image Server URL would be:

```
http://myhost.mydomain.com/cgi-bin/ion-i.exe
```

The ION Script Parser URL would be:

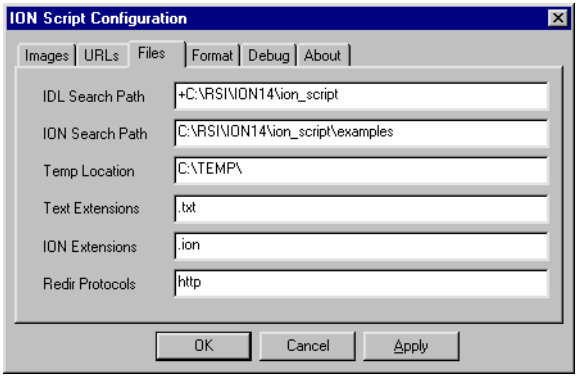
```
http://myhost.mydomain.com/cgi-bin/ion-p.exe
```

Note

If you are using Microsoft IIS, the `.exe` extension must be included in URLs that point to executables. If you get an error informing you that the file or script you requested cannot be found, make sure you are using `ion-p.exe` instead of `ion-p` in any URLs in your HTML and ION Script pages, and in the location field of your browser when requesting a URL.

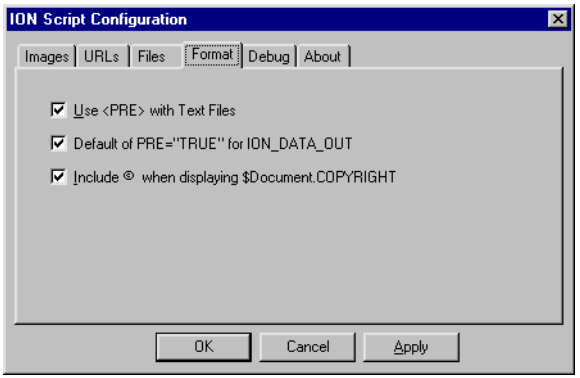
The Files Tab

The Files tab, shown below, is used to specify certain paths, file extensions, and protocols:



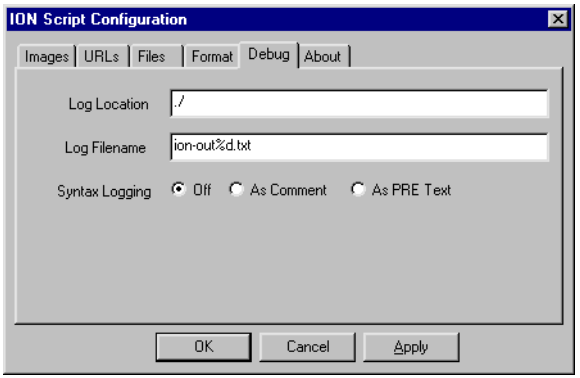
The Format Tab

The Format tab, shown below, is used to specify defaults that affect the way certain text elements are formatted on an ION Script page:



The Debug Tab

The Debug tab, shown below, is used to specify the name and location of the log file created when DEBUG=“TRUE” is specified for the ION_IMAGE or ION_DATA_OUT tag, and the type of error reporting for ION Script syntax errors:



Configuring ION Script for UNIX

The UNIX version of ION Script uses a shell script called `.ionsrc` to define various default settings, preferences, and file paths. The `.ionsrc` file, listed below, is installed in the `ion_script` directory.

Each time the ION Script parser is called, it executes the `.ionsrc` file. The parser first searches for the `.ionsrc` file in the same directory in which the parser itself, `ion-p`, is located (your Web server's `cgi-bin` directory). If the file is not found in the `cgi-bin` directory, it looks for the file in the `ion_script` directory. By copying the `.ionsrc` file to your Web server's `cgi-bin` directory, you can specify different configuration settings for each Web server running ION Script.

To configure ION Script for UNIX, do the following:

1. If desired, copy the `.ionsrc` file from the `ion_script` directory to your Web server's `cgi-bin` directory. If you do this, be sure to edit only the file in the `cgi-bin` directory, as changes you make to the file in the `ion_script` directory will be ignored.
2. Open the `.ionsrc` file in a text editor.
3. If you didn't configure ION Script to use the path to your Web server's CGI executables directory and fully qualified domain name during the installation process, see "Manually Configuring Your UNIX Web Server" on page 16.
4. The `.ionsrc` file consists of a number of settings using the format *option* = *value*. Refer to "ION Script Configuration Options" on page 18 for a description of each option. Modify values in the `.ionsrc` file as desired. Change only the values on the right side of the = sign. Case is important — do not change the case when editing a value. The following is a listing of the `.ionsrc` file:

```
# ION Script Resource File
#
# ION Script Version
VERSION = 6.1

# *** Images ***

# Image Width
Image Width = 320

# Image Height
Image Height = 256
```

```
# Image Label
Image Label = ION Image

# Link Border
Link Border = 1

# Direct Graphics Image Type
GR1 Type = PNG8

# Object Graphics Image Type
GR2 Type = JPEG24

# *** URLs ***

# Image Server URL
Image Server URL = http://hostname/cgi-bin/ion-i

# ION Script Parser URL
Parser URL = http://hostname/cgi-bin/ion-p

# *** Files ***

# IDL Search Path
IDL Path = /usr/local/rsi/ion_6.1/ion_script

# ION Search Path
ION Search Path = .:$ION_DIR/ion_script/examples

# ION Temp Location
ION Temp Location = /tmp

# Text Extensions
Text Exts = txt,text

# ION Extensions
ION Exts = ion

# Redirect Protocols
Redir Protos = http

# *** Format ***

# Use <PRE> with Text Files
Pre Txt = t

# Default of PRE="TRUE" for <ION_DATA_OUT>
Pre DO = t

# Include (c) when displaying $Doc.COPYRIGHT
```

```
Use Copy = t

# *** Debug ***

# Location for debug log files
ION Debug Location = ./

# Format for ion-i log files
ION Debug Filename = ion-out%d.txt

# Syntax errors from ion-p can be returned to the browser.
# Valid options are OFF, COMMENT, or PRE
ION Debug Syntax = OFF
```

Manually Configuring Your UNIX Web Server

If you skipped the “Web Server Configuration” dialog during the UNIX installation process, you will need to either execute the `script_config` script located in the *`RSI-DIR`*`/ion_6.1/ion_script/bin` directory, or perform the following tasks manually to ensure that ION Script knows the necessary information about your Web server.

Copy the ION Script Executables

Web servers have a default directory in which CGI executables are located. This directory is often called `cgi-bin`. For the Apache Web Server, the path to this directory might look something like

```
/usr/local/apache2/cgi-bin
```

1. Copy the files `ion-i` and `ion-p` from the directory *`RSI-DIR`*`/ion_6.1/ion_script/cgi-bin` to your Web server's CGI executables directory. Make sure these files have execute permission for all users.
2. Copy the files `ion-I` and `ion-P` from the directory *`RSI-DIR`*`/ion_6.1/ion_script/cgi-bin/bin.`*`platform`*, where *`platform`* is the name of your platform, to your Web server’s CGI executables directory. Make sure these files have execute permission for all users.

Edit the Image Server and Parser URLs

1. Open the file *`RSI-DIR`*`/ion_6.1/ion_script/.ionsrc` in a text editor. This is the ION Script configuration file. If multiple Web servers will be using the same ION Script installation, copy `.ionsrc` to your Web server’s CGI executables directory and modify the file in that location.

2. Edit the Image Server URL and ION Script Parser URL entries so they contain the fully qualified domain name of your Web server, and the correct path to your Web server's CGI executables directory. For example, the Image Server URL might look like the following:

   ```
   http://myhost.mydomain.com/cgi-bin/ion-i
   ```

3. Save the `.ionsrc` file.

ION Script Configuration Options

The following configuration options are available in ION Script. The options are grouped as they are on the tabs of the Windows configuration program, `IONScriptConfig.exe`, and in the UNIX resource file `.ionsrc`.

Images

Width, Height

The Width and Height fields are used to specify the default dimensions, in pixels, of images created with the ION_IMAGE tag. These default values can be overridden by specifying the WIDTH and HEIGHT attributes of the ION_IMAGE tag.

Direct Graphics Image Type

The Direct Graphics Image Type specifies the image format for images created with ION_IMAGE with the TYPE attribute set to DIRECT, which causes the image to be created using IDL's Direct Graphics. This default value can be overridden by specifying the IMG_TYPE attribute of the ION_IMAGE tag. See "Graphics in ION Script" on page 91 for a discussion of how to choose the proper image format.

Object Graphics Image Type

The Object Graphics Image Type specifies the image format for images created with ION_IMAGE with the TYPE attribute set to OBJECT, which causes the image to be created using IDL's Object Graphics. This default value can be overridden by specifying the IMG_TYPE attribute of the ION_IMAGE tag. See "Graphics in ION Script" on page 91 for a discussion of how to choose the proper image format.

Label

The Label field is used to specify the default text displayed in place of an image for browsers that do not support images, and for browsers in which the automatic display of in-line images has been turned off. This default value can be overridden by specifying the LABEL attribute of the ION_IMAGE tag.

This label is also used in some cases as the tooltip that appears when you hold the mouse cursor over the image, depending on which browser type and version you are using. Because different browsers (and different versions of each) implement alternate text and tooltips differently, it helps to know how ION Script maps the label to HTML. This label value (or the LABEL attribute of ION_IMAGE, if specified) becomes the value of the ALT attribute of the HTML <IMG> tag for an ION_IMAGE without an EVENT, or the ALT attribute of the

<INPUT TYPE="IMAGE"> tag for an ION_IMAGE with an EVENT. Because some browsers handle images created with the <IMG> tag differently than those created with the <INPUT> tag, the name of the EVENT may be displayed in place of the value specified for the label if the EVENT attribute of the ION_IMAGE tag has been specified for an image.

Border

The Border field is used to specify the default thickness of the border drawn around an image created with ION_IMAGE or an ION_BUTTON of type IMAGE. A setting of 0 causes images to be drawn without a border. This default value can be overridden by specifying the BORDER attribute of the ION_IMAGE or ION_BUTTON tag.

Note

<ION_BUTTON TYPE="IMAGE"> creates <INPUT TYPE="IMAGE">. Note that not all browsers support borders on <INPUT TYPE="IMAGE">.

When the EVENT attribute has been specified for the ION _IMAGE tag, the border is drawn in the color specified by the LINK attribute of the ION_BODY tag, if used, or in the link color used by the browser. If no EVENT has been specified, the border color is black. If desired, you could wrap the ION_IMAGE tag in a <FONT> tag to specify a different color as follows:

```
<FONT COLOR="#808080">
<ION_IMAGE> ... </ION_IMAGE>
</FONT>
```

URLs

Image Server URL

The Image Server URL field is used to specify the path to the ION Script Image Server, `ion-i`. This default value can be overridden by specifying the SERVER attribute of the ION_IMAGE tag. See “ION_IMAGE” on page 159.

ION Script Parser URL

The ION Script Parser URL field is used to specify the path to the ION Script parser, `ion-p`.

Files

IDL Search Path

The IDL Search Path field is used to specify the search path used by IDL for `.pro` and `.sav` files. To specify multiple directories, separate each directory with a semicolon (Windows) or a colon (UNIX). Place the "+" symbol at the beginning of a directory to indicate that all subdirectories of the specified directory should be searched. For example, the following Windows IDL Search Path specifies that the directory `C:\rsi\IDL61\products\ION61\ion_script\examples` and all its subdirectories be searched, as well as the directory `C:\ion\pro`:

```
+C:\rsi\IDL61\products\ION61\ion_script\examples;C:\ion\pro
```

The following UNIX IDL Search Path specifies that `/usr/local/rsi/ion` be searched, as well as the directory `/home/ion` and all its subdirectories:

```
/usr/local/rsi/ion:+/home/ion
```

Note

By default, the directory `\`*`RSI-DIR`*`\IDL61\products\ION61\ion_script\examples` (Windows) or */*`RSI-DIR`*`//ion_6.1/ion_script/examples` (UNIX) and its subdirectories are automatically searched by IDL. You do not need to add this directory to the IDL Search Path.

Note

On UNIX, make sure that any directory you add to the IDL Search Path has the proper read permissions.

See "Where to Store Your Files" on page 38 for strategies on locating IDL files.

ION Search Path

The ION Search Path field is used to specify the search path for URLs using the file:// or ion:// protocols. ION Script searches the ION Search Path for the specified file specified by the "page" parameter in the URL. When ION Script searches the ION Search Path, it will search all directories specified in this field in the order listed. Multiple directories can be specified by separating the directory names with a semicolon (Windows) or colon (UNIX), as in the following examples:

Windows:

```
C:\rsi\IDL61\products\ION61\ion_script\examples;C:\ion;C:\ion\test
```

UNIX:

```
/usr/local/rsi/ion_6.1/ion_script/examples/:
    /home/ion:/home/ion/test
```

Note

You cannot use the "+" symbol in the ION Search Path to indicate that all subdirectories of the specified directory should be searched. You must explicitly specify each directory to be searched.

If you specify an absolute path in your URL, such as

```
http://myserver/cgi-bin/ion-p?page=c:\ion\myfile.ion)
```

ION Script attempts to execute the specified page and does not search the ION Search Path. See "Specifying URLs" on page 70 and "Absolute Paths vs. the ION Search Path" on page 72 for more on URLs and the ION Search Path. See "Where to Store Your Files" on page 38 for strategies on locating your `.ion` and `.html` files.

Temp Location

Set this field to the path specifying the location of the temporary graphics and text files that ION Script creates. If not specified, UNIX servers will write to `/tmp`, and Windows servers will write to the location specified by the TEMP system environment variable. The value of this setting can be accessed programmatically via the $ION.temp system variable.

Note

If the directory you have specified does not exist, or if the web server does not have permissions to write to the directory, images created with the ION_IMAGE tag do not display, and ION Script does not display any errors. Make sure that the temp directory you have specified exists and the web server has permissions to write to the temp directory.

Text Extensions

The Text Extensions field is used to specify the file extensions that are to be considered text files by ION Script. Separate multiple extensions by commas, such as

```
.txt,.text
```

You can omit the period:

```
txt,text
```

Any extension listed in the Text Extensions field can be used in place of `xxx` in the following cases, and the files will be parsed as text files:

- When included with ION_INCLUDE:

```
<ION_INCLUDE SRC="filename.xxx"/>
```

- When an EVENT points to a text file as in

```
<EVENT_DECL NAME="event" ACTION="myfile.xxx"/>
```

- Whenever a URL of the following format is used:

```
http://server/cgi-bin/ion-p?page=filename.xxx
```

- When an ION Script page is the action of an HTML form:

```
<FORM NAME="MyForm" ACTION="http://server/cgi-bin/ion-p" >
   <INPUT TYPE="HIDDEN" NAME="page" VALUE="filename.xxx" >
</FORM>
```

See "ION_INCLUDE" on page 165 for an example that shows how text files are handled by ION Script. See "HTML Forms vs. ION Script Forms" on page 74 for more on making ION Script pages the action of an HTML form.

Note

If the ion:// protocol is used, the file is parsed as an ION Script file no matter what extension the file uses.

ION Extensions

The ION Extensions field is used to specify the file extensions that are to be considered ION Script files by ION Script. Separate multiple extensions by commas, such as

```
.ion,.xyz
```

You can omit the period:

```
ion,xyz
```

Any extension listed in the ION Extensions field can be used in place of *xxx* in the following cases, and the files will be parsed as ION Script files:

- When included with ION_INCLUDE:

```
<ION_INCLUDE SRC="protocol://filename.xxx"/>
```

- When an EVENT points to an ION Script file:

```
<EVENT_DECL NAME="event" ACTION="myfile.xxx"/>
```

- Whenever a URL of the following format is used:

```
http://server/cgi-bin/ion-p?page=filename.xxx
```

- When an ION Script page is the action of an HTML form:

```
<FORM NAME="MyForm" ACTION="http://server/cgi-bin/ion-p" >
   <INPUT TYPE="HIDDEN" NAME="page" VALUE="filename.xxx" >
</FORM>
```

  See "HTML Forms vs. ION Script Forms" on page 74 for more on making ION Script pages the action of an HTML form.

Note

If the ion:// protocol is used, the file is parsed as an ION Script file no matter what extension the file uses.

Redir Protocols

When ION Script encounters a URL that uses a protocol specified in the Redir Protocols field, it will redirect to the URL rather than attempt to load the URL. Redirecting to a Web page is usually quicker than loading the page, and when a URL is outside a firewall, it may not be possible to connect to the site. In this case, a redirect allows you to load the URL. The URLs to which this applies include:

- EVENT actions
- URLs that use the query string format (?page=*url*)
- URLs specified as the VALUE of a hidden form field with NAME="page" in an HTML form (when the ACTION of the form is the ION Script parser). See "HTML Forms vs. ION Script Forms" on page 74 for more on making ION Script pages the action of an HTML form.

Format

Use <PRE> with Text Files

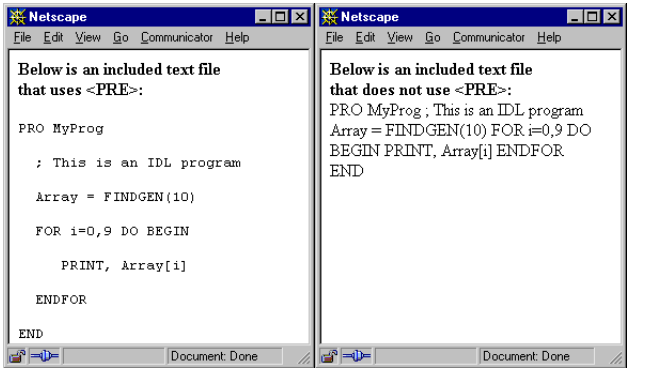
Select this checkbox on Windows, or set "Pre Txt = t" in the `.ionsrc` file on UNIX, if you want text files parsed by ION Script to be rendered using the HTML <PRE> tag. Text files are parsed by ION Script in the following cases:

- When included with <ION_INCLUDE SRC="*filename*.txt"/> (or other extensions defined in the Text Extensions field)
- When an EVENT points to a text file
- Whenever a URL of the following format is used:

```
http://server/cgi-bin/ion-p?page=filename.xxx
```

  where xxx is any extension listed in the Text Extensions field.

The following figure shows the difference between included text files when this feature is turned on and off:



For more on including text files, see “ION_INCLUDE” on page 165.

Default of PRE="TRUE" for <ION_DATA_OUT>

Select this checkbox on Windows, or set “Pre DO = t” in the `.ionsrc` file on UNIX, to use TRUE as the default value of the PRE attribute of the ION_DATA_OUT tag. See “ION_DATA_OUT” on page 141 for a discussion and example using the PRE attribute.

Include © when displaying $Document.COPYRIGHT

Select this checkbox on Windows, or set “Use Copy = t” in the `.ionsrc` file on UNIX, to display the © symbol when using the $Document.COPYRIGHT system variable. See “COPYRIGHT” on page 157 and “$Document” on page 51 for more on including a copyright statement in an ION Script page.

Debug

ION Debug Location

Set this field to a writable path specifying the location on the server machine in which to write the debug log file. The debug log file is created when the DEBUG attribute of either the ION_DATA_OUT or ION_IMAGE tag is set to TRUE. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL. If not set, the file is written to the default location that your Web server uses for CGI executables. For example, on the Apache Web server, this would be the `cgi-bin` directory.

ION Debug Filename

Set this field to the filename of the debug log file. The debug log file is created when the DEBUG attribute of either the ION_DATA_OUT or ION_IMAGE tag is set to TRUE. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL. Instead of using a specific filename, you can include the %d formatting string in it, such as “ioni-log%d.txt”. This causes the process ID of `ion-i` to be inserted where the formatting characters are. Because the process ID is different each time `ion-i` is requested, the log filename is different each time, thereby preventing the log file from being overwritten. If not set, the default filename is “ion-out%d”.

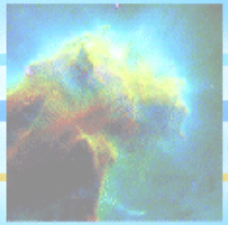
Syntax Logging

This field defines the default method for reporting ION Script syntax errors. On Windows, select one of the following buttons:

- Off — Error messages are not reported.
- As Comment — Error messages are included in the returned HTML page as part of an HTML comment. This makes the error messages accessible but invisible to the client. To view the error messages, you must view the source of the HTML page sent to the browser by selecting **View → Page Source** (Netscape) or **View → Source** (Internet Explorer).
- As PRE Text — Error messages are included in the returned HTML page as <PRE> text, and will appear at the bottom of the returned HTML page. This is useful during development of your ION Script applications.

On UNIX, set the “ION Debug Syntax” value to either “OFF,” “COMMENT,” or “PRE” in the `.ionsrc` file.

The method selected in the ION Script Configuration Utility or `.ionsrc` file becomes the default method for error reporting. This default method can be overridden on a page by page basis by setting the SYNTAX attribute for the <ION_SCRIPT> tag to either “OFF,” “COMMENT,” or “PRE.”



Chapter 2:
Overview

This chapter provides a high-level overview of ION Script. The following topics are covered in this chapter:

- What Is ION Script?
- Requesting an ION Script Page
- Where to Store Your Files
- Running the ION Script Examples

What Is ION Script?

ION Script is a language for creating Web-based IDL visualization and analysis applications. It contains tags, similar to HTML tags, for embedding IDL visualizations and data in a Web page. It has a rich set of variable, event, and flow control tags that allow you to create powerful, interactive Web applications.

ION Script is lightweight, easy to learn and implement, can capable of creating applications that range in complexity from simple Web-based visualizations to interactive data-picking applications, including Region-Of-Interest selection (to drill down through a data set or zoom in on an image, for example). ION Script provides a mechanism for the developer to track the client's state across multiple Web pages.

ION Script can be easily integrated with other Web-based technologies such as JavaScript, ActiveX, and database applications, and can be easily incorporated into existing Web pages.

If you require the ability to expose your IDL data and visualizations to Internet or intranet users, ION Script is the perfect tool. For cases in which the Web browser is a suitable container for your application, ION Script provides the necessary functionality without burdening you with the complexity of Java programming, or burdening the users of your application with the overhead required to run a Java application.

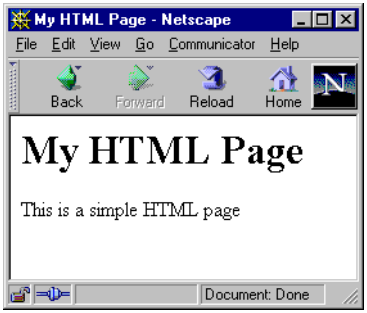
This brief overview of ION Script will introduce you to some of the main elements of the ION Script language before we get into the more specific details of writing ION Script applications. After reading this section, you'll have a general idea of what an ION Script page looks like and the kinds of applications for which ION Script can be used. For details on the syntax and usage of each ION Script tag, refer to Chapter 5, "ION Script Tag Reference".

ION Script Pages

An ION Script page is similar in organization to an HTML page. In fact, ION Script pages can contain HTML. Consider the following HTML page:

```
<HTML>
<HEAD>
   <TITLE>My HTML Page</TITLE>
</HEAD>
<BODY>
   <H1>My HTML Page</H1>
   This is a simple HTML page
</BODY>
</HTML>
```

This simple page is rendered by the browser as follows:


My HTML Page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home
My HTML Page
This is a simple HTML page
Document: Done

ION Script pages are constructed in much the same way:

```
<ION_SCRIPT>

<ION_HEADER>
   <TITLE>My ION Script Page</TITLE>
</ION_HEADER>

<ION_BODY>
   <H1>My ION Script Page</H1>
   This is a simple ION Script page
</ION_BODY>

</ION_SCRIPT>
```

Every ION Script page must have the ION_SCRIPT and ION_BODY tags. The ION_SCRIPT tag is a container for everything in the ION Script page. All content on the page must reside inside the ION_SCRIPT block. Everything inside the ION_BODY tag is converted to HTML to create the Web page delivered to the user. Optionally, an ION Script page can contain an ION_HEADER block, used to define document attributes such as the title, to declare ION Script variables and events, and to include any HTML tags you would use inside the HEAD element of an HTML page, such as <META>, <SCRIPT>, and <STYLE>.

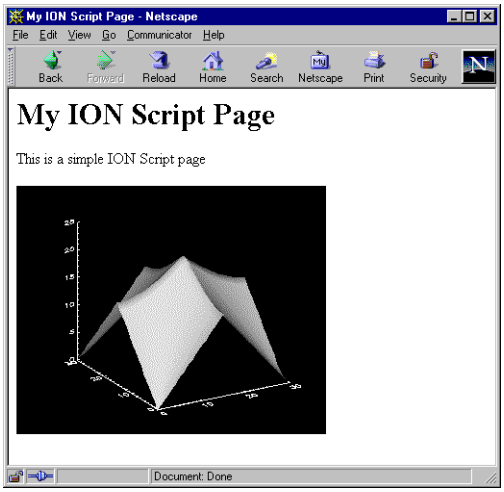
This ION Script page is parsed by the ION Script parser to create the following HTML page:

My ION Script Page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search
My ION Script Page
This is a simple ION Script page
Document: Done

Creating the above page with ION Script wouldn't make much sense because the same page could be created with pure HTML. But pure HTML doesn't provide the means to, for example, embed a dynamically-generated IDL image in a Web page. By adding a couple of simple tags, ION_IMAGE and IDL, we can easily accomplish this with ION Script:

```
<ION_SCRIPT>

<ION_HEADER>
   <TITLE>My ION Script Page</TITLE>
</ION_HEADER>

<ION_BODY>
   <H1>My ION Script Page</H1>
   This is a simple ION Script page<BR><BR>
   <ION_IMAGE TYPE="DIRECT">
      <IDL>
         shade_surf, dist(30)
      </IDL>
   </ION_IMAGE>
</ION_BODY>

</ION_SCRIPT>
```

This code gives us the following page:


My ION Script Page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
My ION Script Page
This is a simple ION Script page
Document: Done

The header section of an ION Script page is also where we define variables and events, using the VARIABLES, VARIABLE_DECL, EVENTS, and EVENT_DECL tags. The following example creates a button that triggers the PLOT event, which executes the page `plot.ion`:

```
<ION_SCRIPT>
<ION_HEADER>
   <TITLE>My ION Script Page</TITLE>

   <VARIABLES>
      <VARIABLE_DECL NAME="A" VALUE="5" TYPE="INT"/>
   </VARIABLES>

   <EVENTS>
      <EVENT_DECL NAME="PLOT" ACTION="ion://plot.ion"/>
   </EVENTS>
</ION_HEADER>

<ION_BODY>
   <ION_FORM>
      <ION_BUTTON EVENT="PLOT" TYPE="BUTTON" LABEL="Draw Plot"/>
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

Variables and events allow you to create interactive applications in which the user can control the behavior of the application and the appearance of generated images and data.

Also important in creating interactive applications are ION Script's flow control tags, ION_IF, ION_ELSE, and ION_ELSEIF. These tags allow you to perform certain actions depending on the actions of the user, such as reporting invalid entries:

```
<ION_IF EXPR="$USERENTRY EQ 0">
   You can't enter 0!
<ION_ELSE/>
   You entered <ION_VARIABLE NAME="$USERENTRY"/>
</ION_IF>
```

As with HTML tags, some ION Script tags are called *tag pairs*, meaning that they must contain an opening and closing tag, such as ION_BODY:

```
<ION_BODY>
</ION_BODY>
```

Other ION Script tags are called *single tags*, which do not contain a closing tag. Unlike single HTML tags, such as <BR> and <INPUT>, single ION Script tags such as VARIABLE_DECL use XML syntax and are therefore closed with "/>" instead of just ">":

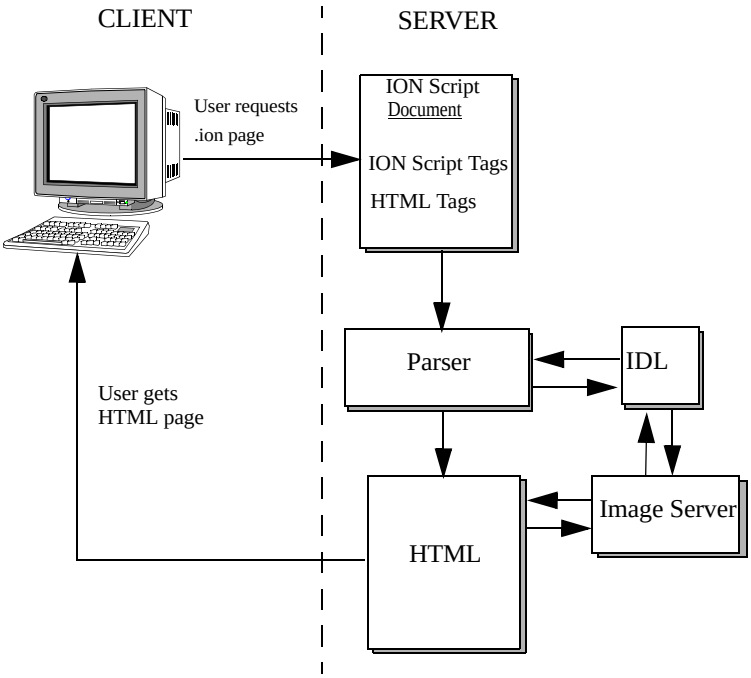
```
<VARIABLE_DECL NAME="A" VALUE="5" TYPE="INT"/>
```

ION Script Applications

An ION Script application is composed of one or more ION Script pages that contain ION Script and HTML tags. An ION Script application can also contain pure HTML pages that can link to other ION Script pages. Unlike an HTML page, which is passed directly from the Web server to the client, an ION Script page is first parsed by the ION Script parser, which converts any ION Script content to HTML before returning the page to the client. The ION Script content can consist of tags that embed IDL data or visualizations, and tags that add "brains" to your application, allowing you to generate Web pages dynamically depending on the user's actions. For example, you can allow your users to click on an IDL-generated image to zoom in on a certain part of the image, or provide users with a choice of image processing algorithms to apply to the image.

Anatomy of an ION Script Application

Before you learn the details of writing an ION Script application, it helps to understand what ION Script does under the hood when a client requests an ION Script page. ION Script tags are processed by the ION Script parser and converted to HTML. As illustrated in the following figure, the output of every ION Script application is an HTML page:



The following sequence of events occurs when a request is made for an ION Script page:

1. The client requests an ION Script page by requesting a URL of the form `http://myserver/cgi-bin/ion-p?page=mypage.ion`, by clicking a link pointing to such a URL, or by submitting an HTML form with an action that points to the ION Script parser.
2. The Web server passes the request to the parser (the CGI program `ion-p`).
3. The parser converts ION Script content to HTML:

- The content of ION_DATA_OUT tags is sent to the Data Server, which passes the IDL content to IDL for processing, and the Data Server returns the IDL output to the Web page.
- Any other ION Script content, such as ION_EVALUATE and ION_VARIABLE, is evaluated by the parser and converted to HTML.
- HTML tags are passed as is to the outputted HTML page.
- ION_IMAGE tags are converted to HTML <IMG> tags. These <IMG> tags are then processed by the Image Server, which passes the IDL content to IDL for processing, and the Image Server returns the IDL-generated image to the Web page as a PNG or JPEG image.
- Any attribute used inside an ION Script tag that is not recognized by the ION Script Parser as an ION Script attribute is passed unchanged to the browser. This allows you to use in an ION Script tag any valid attribute of the HTML tag to which the ION Script tag is converted. For each ION Script tag listed in Chapter 5, “ION Script Tag Reference”, there is a section called “HTML Mapping” that tells you what HTML, if any, is produced by that tag.

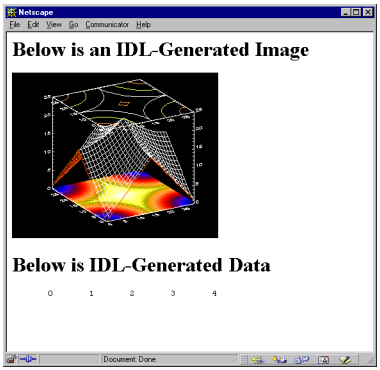
Example

Suppose you have the following ION Script page, `mypage.ion`, on your Web server:

```
<ION_SCRIPT>
<ION_BODY>
   <H1>Below is an IDL-Generated Image</H1>
   <ION_IMAGE IMG_TYPE="PNG8" TYPE="DIRECT">
      <IDL>
         LOADCT, 5
         SHOW3, DIST(30)
      </IDL>
   </ION_IMAGE><BR>
   <H1>Below is IDL-Generated Data</H1>
   <ION_DATA_OUT>
      <IDL>
         Array=INDGEN(5)
         Print, Array
      </IDL>
   </ION_DATA_OUT>
</ION_BODY>
</ION_SCRIPT>
```

If you’re familiar with HTML, you’ll notice a couple plain HTML tags in this page: the <H1> and <BR> tags. These HTML tags are sent directly to the HTML page that the parser creates as soon as they are encountered. The other tags are ION Script tags,

which, with the help of the Image Server and IDL, are converted to HTML. When the client requests `mypage.ion`, the following page is returned:



If you view the source code returned to the client, you'll see that it looks very different from the original ION Script page that was requested:

```
<HTML>
<BODY>
<H1>Below is an IDL-Generated Image</H1>
<IMG NAME="IMG" ALT="ION Image" WIDTH="320" HEIGHT="256"
BORDER="1" SRC="http://www.myserver.com/cgi-bin/ion-
i?DATA=DB1AA6B14FAC62CBF4BDA47CBDE4C4C5E2649F910E046ACB07A2B910FA4
FA4FA9E27B062466F914D567636B09190499CBC3871A0A7028F8DDC19643548CC9
5233B8B5DDF22128893F798DAC895C51530E412C60F77F58162F0">
<BR>
<H1>Below is IDL-Generated Data</H1>
<PRE>       0       1       2       3       4
</PRE>
</BODY>
</HTML>
```

Note the following with regard to this HTML code:

- The <HTML> and <BODY> tags are automatically added by ION Script.
- The <H1> and <BR> tags are included as is.
- The ION_IMAGE tag becomes an HTML <IMG> tag with a SRC attribute that points to the ION Script Image Server. The URL specified for the SRC attribute contains encrypted information used to generate the image.
- The ION_DATA_OUT tag becomes plain text, surrounded in this case by the HTML <PRE> tag by default.

Requesting an ION Script Page

In the previous section, we skipped over the details of how the client requests the ION Script page in the first place. When requesting an HTML page, you normally request an HTML file using a URL such as

```
http://www.myserver.com/mypage.html
```

When requesting an ION Script page, you are requesting a CGI application—the ION Script parser—and providing the name of the page to pass to the parser. When calling CGI applications via a URL, you pass information to the CGI program by encoding the information in the URL. This encoded information is called a *query string*. The ION Script parser takes one argument: the name of the file to be parsed. The URL you use to request an ION Script page takes the following form:

```
http://servername/cgi-bin/ion-p?page=pagename.ion
```

where *servername* is the fully-qualified domain name of the server on which the ION Script parser is located, `ion-p` is the name of the ION Script parser, and `pagename.ion` is the name of the ION Script page. Note that this URL assumes that the ION Script parser is located in the `cgi-bin` directory of the server. Different Web servers use different names for this directory, therefore, the path to `ion-p` may differ for your Web server. Also, you must specify the ION Script Parser URL on the URLs tab of the configuration utility. See "ION Script Parser URL" on page 19.

This URL format is used to load ION Script pages in the following cases:

- When you type the URL of an ION Script page in your browser
- For the HREF attribute of the <A> tag when creating a hyperlink to an ION Script page
- For the SRC attribute of the <FRAME> or <IFRAME> tag when the frame source is an ION Script page

It is also possible to include name/value pairs in your URL to pass parameters to the requested page. For example, you could pass the DATA parameter with a value of 5 to the page `mypage.ion` as follows:

```
http://servername/cgi-bin/ion-p?page=mypage.ion&DATA=5
```

You could then access the DATA parameter in the `mypage.ion` using $Form.DATA.

ION Script pages can also be loaded as the action of an HTML form. HTML forms can use hidden form fields to pass name/value pairs to a CGI application. Instead of including the query string in the ACTION attribute of the <FORM> tag, we create a

hidden field with the NAME attribute set to "page" and the VALUE attribute set to the URL of the ION Script page, as in the following example:

```
<HTML>
<FORM NAME="MyForm" ACTION="http://server/cgi-bin/ion-p">
   <INPUT TYPE="HIDDEN" NAME="page" VALUE="filename.xxx">
   <INPUT TYPE="SUBMIT">
</FORM>
</HTML>
```

Where to Store Your Files

This section provides some strategies for locating your `.ion`, `.html`, and `.pro` files.

How HTML Files Are Normally Requested

You're probably used to requesting HTML files using URLs that look like this:

```
http://www.mydomain.com/index.html
```

In this case, the file `index.html` is located in the Web server's default document directory. On the Apache Web Server, for example, this is called `htdocs` by default. Files requested via the HTTP protocol must be located in or under the `htdocs` directory, unless you have changed the Apache server settings (in the `httpd.conf` file) to include other directories. You could create a subdirectory of `htdocs` called `htmlfiles`, and locate your `.html` files in this directory. The URL to which you would refer would be

```
http://www.mydomain.com/htmlfiles/index.html
```

Often, users request a URL such as

```
http://www.mydomain.com
```

In this case, the Web server will serve a default page, which is specified in the Web server configuration file. If you want your users to be able to access your ION Script application using a URL such as `http://www.mydomain.com`, you need to define a default page in your Web server configuration file, and use this as the name of your application's starting page.

ION Script Files

The ION Search Path is the path ION Script uses to find files specified using URLs of the following format:

```
http://myhost.mydomain.com/cgi-bin/ion-p?page=myfile.ion
```

Initially, `RSI-DIR/IDL61/products/ION61/ion_script/examples` is the only directory specified for the ION Search Path, but you'll probably want to keep your own source files separate from the example files, so you may wish to create your own directory in which to store ION Script files. For example, you could create a directory called `C:\ion` (Windows) or `/home/ion` (UNIX) in which to store your `.ion` files. Note that this directory must be accessible from your Web server. You would then add this directory to your ION Search Path, which would then look like this:

Windows: `C:\rsi\IDL61\products\ION61\ion_script\examples;C:\ion`

UNIX: `.:/usr/local/rsi/ion_6.1/ion_script/examples:/home/ion`

Keeping .ion and .html Files in the Same Place

For your ION Script applications, you may wish to keep all your `.ion` and `.html` files in the same place. To avoid having to move your HTML files to your Web server's default documents directory, you can load HTML pages using the same URL format as you use to load ION Script pages. For example, if the file `myfile.html` is located in your ION Search Path rather than the `htdocs` directory of your Web server, you could load the page using a URL of the following format:

```
http:///myhost.mydomain.com/cgi-bin/ion-p?page=myfile.html
```

See "Overview of Fixed Framesets" on page 82 for an example of how the SRC attribute of the <FRAME> tag can also use this type of URL.

If you want to keep your `.ion` and `.html` files in the same directory, but also want to use your Web server's default documents directory to store your `.html` files, you could add the Web server's default documents directory to your ION Search Path.

IDL Files

The IDL Search Path is the path IDL uses to find `.pro` and `.sav` files. Initially, `\`*`RSI-DIR`*`\IDL61\products\ION61\ion_script` and its subdirectories are the only directories specified for the IDL Search Path.

Note

By default, the directory `\`*`RSI-DIR`*`\IDL61\products\ION61\ion_script\examples` (Windows) or `/`*`RSI-DIR`*`/ion_6.1/ion_script/examples` (UNIX) and its subdirectories are automatically searched by IDL. You do not need to add this directory to the IDL Search Path.

However, you may want to keep your own IDL files in another location. For example, you could create a directory called `C:\ion\pro` (Windows) or `/home/ion/pro` (UNIX) in which to store your `.pro` and `.sav` files. You would then add this directory to your IDL Search Path, which would then look like this:

Windows:

```
+C:\rsi\IDL61\products\ION61\ion_script\examples;+C:\ion\pro
```

UNIX:

```
+/usr/local/rsi/ion_6.1/ion_script/examples:
+/home/ion
```

You can now store all `.pro` and `.sav` files in this new directory.

Running the ION Script Examples

Several example applications are installed with ION Script. These examples are located in the *RSI-DIR*\IDL61\products\ION61\ion_script\examples (Windows) or */RSI-DIR*/ion_6.1/ion_script/examples (UNIX), where *RSI-DIR* is the directory in which you installed ION Script. The page index.ion contains links to three different examples pages:

- Basic Examples — These examples illustrate simple ION Script concepts. Examples include how to create a simple ION Script page, how to define variables and events, how to use form elements such as buttons, checkboxes, and radio buttons, and how to use HTML frames.
- Advanced Examples —These more complex examples demonstrate some of the advanced programming concepts in ION Script, such as image processing, form validation using JavaScript, and using IDL to dynamically create HTML tables and droplists.
- Demos — This link takes you to the RSI ION site, www.RSInc.com/ion, where you can view interactive Web applications built on ION Script.

Assuming the path to your cgi-bin directory is /cgi-bin, you can load the ION Script examples page using the following URL, where *servername* is the name of your server:

```
http://servername/cgi-bin/ion-p?page=index.ion
```

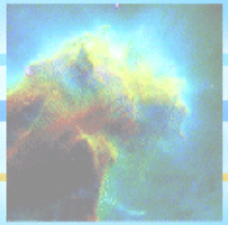
You can also find a link to index.ion by requesting the page called about as follows:

```
http://servername/cgi-bin/ion-p?page=about
```

This page lists the current version of ION Script, and contains a link to the examples page index.ion.

Note

If you are using Microsoft IIS, the .exe extension must be included in URLs that point to executables. If you get an error informing you that the file or script you requested cannot be found, you may need to use ion-p.exe instead of ion-p in any URLs in your HTML and ION Script pages, or in the location field of your browser when requesting a URL. All examples in the examples directory were modified during your ION 6.1 installation for Windows.



Chapter 3:
Variables, Expressions & Operators

This chapter discusses the core language elements of ION Script. The following topics are covered in this chapter:

- Variables
- System Variables
- Expressions
- Operators

Variables

This section discusses how to declare and use ION Script variables, the types of data an ION Script variable can hold, and how to create persistent variables that can be used across multiple ION Script pages.

All ION Script variables must be prefixed by the $ symbol when they are referenced, and when they are assigned a value using the ION_EVALUATE tag. They are *not* prefixed with the $ symbol, however, when they are declared using the VARIABLE_DECL tag.

Declaring & Assigning Values

ION Script variables can be declared using the <VARIABLE_DECL> tag as follows:

```
<VARIABLE_DECL NAME="name" TYPE="type" VALUE="value"
   PERSIST="TRUE"/>
```

Declaring a variable allows you to explicitly define the type of data held by the variable, and allows you to make the variable persistent.

Although it is recommended, ION Script variables do not need to be declared before they are used. Variables can be assigned values equal to the result of an expression using the ION_EVALUATE tag as follows:

```
<ION_EVALUATE EXPR="$variable = expression" />
```

Variable Types

ION Script currently uses four data types:

- DOUBLE — Double precision
- INT — Integer
- BOOL — Boolean
- STR — String

ION Script variables are strongly typed, meaning the type of data they hold cannot change once they have been declared or assigned a value. Variables are assigned a data type only once: if you declare the variable using VARIABLE_DECL, then you assign the type using the TYPE attribute. If you assign a value to the variable using ION_EVALUATE, then the type assumed by the variable depends on the result of the expression to which it is assigned. Reassigning a value to a variable that has already been declared or assigned a value does not change the type of the variable, even if the value that it is reassigned is of a different type than the variable.

If you assign a value to an undeclared variable, the variable assumes one of the following types:

- If you assign a numeric value to an undeclared variable, the type becomes double.
- If you assign a string value to an undeclared variable, the type becomes string. For example, the following code creates a string variable and assigns the value 'Smith':

```
<ION_EVALUATE EXPR="$Name='Smith'" DISPLAY="TRUE"/>
```

- If you assign TRUE or FALSE to an undeclared variable, the type becomes boolean. For example, the following code creates a boolean variable and assigns a value of true:

```
<ION_EVALUATE EXPR="$Selected=TRUE" DISPLAY="TRUE"/>
```

The only case in which ION Script will convert a value from one type to another is when you declare a variable as a TYPE that is different from the value to which you assign it. Consider the following examples:

- If you declare a variable as an INT equal to 1.666, the value is converted to an integer equal to 1. (When ION Script converts the floating-point value to an integer, the decimal portion is truncated.)
- If you declare a variable as an INT and assign it a non-numeric string value, the value becomes an integer equal to 0.
- If you declare a variable as a DOUBLE and assign it a non-numeric string value, the value becomes a double-precision value equal to 0.00.
- If you declare a variable as a STR and assign it a number, the number is converted to double-precision and then to a string. To avoid having the number converted to double before string, nest single quotation marks inside the double quotation marks as follows:

```
<VARIABLE_DECL NAME="A" TYPE="STR" VALUE=" '35.4' " />
```

- If you declare a variable as a BOOL equal to a number, the value becomes false.

A single expression can contain variables and/or values of different types. The result of an expression containing variables and/or values of multiple types assumes the type of the highest-precision value. For example, adding an INT to a DOUBLE results in a DOUBLE. The degree of precision, from highest to lowest, is DOUBLE, INT, BOOL, STR. When the expression contains a string added to a numeric value (using the + operator), the result is a string.

Assigning String Literals to ION Script Variables

Whenever you assign a string literal as the value of an ION Script variable, the string literal should be enclosed in single quotation marks, which are then enclosed in the double quotation marks used to delimit the attribute value. For example, the following code shows the correct way to assign a string literal to the VALUE attribute of the VARIABLE_DECL tag:

```
<VARIABLE_DECL NAME="animal" TYPE="STR" VALUE="'dog'"/>
```

If you are assigning a string variable as opposed to a string literal to the VALUE attribute, do not enclose the variable in single quotation marks. For example, if the variable animal has already been defined as a string, you would assign the value of the animal variable to a new variable as follows:

```
<VARIABLE_DECL NAME="pet" TYPE="STR" VALUE="$animal"/>
```

Determining Variable Type

To easily determine the data type of a variable, use the ISTYPE operator as follows:

$*VarName* ISTYPE *VarType*

where *VarName* is the name of the variable, and *VarType* is one of the following string literals: 'BOOL', 'STR', 'INT', 'DOUBLE', or 'UNDEF'.

Note

The left side of the ISTYPE expression must be a variable and not an expression. If you need to determine the resulting type of an expression, assign the expression to a variable and use ISTYPE to determine the type of the variable.

For example, to determine whether the variable `MyVar` is undefined, you could use the following ION Script code:

```
<ION_IF EXPR="$MyVar ISTYPE 'UNDEF'">
   MyVar is undefined
<ION_ELSE/>
   The value of MyVar is <ION_VARIABLE NAME="$MyVar"/>
</ION_IF>
```

Null Strings Versus Undefined Strings

There is a difference between an empty (null) string and an undefined string. A null string is a defined variable that is assigned no characters. If the value you assign to a variable is a null string, an undefined variable, or an expression that cannot be evaluated, and the TYPE is "STR", that variable is defined, and the variable’s value is

a null string. In the following example, the variable A is assigned a null string, and the variable B is not declared:

```
<ION_SCRIPT>
   <ION_HEADER>
      <VARIABLES>
         <VARIABLE_DECL NAME="A" TYPE="STR" VALUE="''"/>
      </VARIABLES>
   </ION_HEADER>
   <ION_BODY>
      <ION_VARIABLE NAME="$A"/>
      <ION_VARIABLE NAME="$B"/>
      <ION_IF EXPR="$A ISTYPE 'UNDEF'">
          A is undefined
      </ION_IF>
      <ION_IF EXPR="$B ISTYPE 'UNDEF'">
         B is undefined
      </ION_IF>
   </ION_BODY>
</ION_SCRIPT>
```

The result of this ION Script page is:

Error: Variable $B does not exist
B is undefined

Variable Names

The following rules apply to variable names in ION Script:

- When referencing the variable, or assigning it a value using ION_EVALUATE, the variable name must be prefixed with the $ symbol.
- The first character (after the $ symbol) must be a letter. Subsequent characters can be letters, numbers, or underscores (_).
- Variable names are case-sensitive. If you declare a variable as $A, you cannot refer to that variable as $a.
- The variable name cannot be one of the following ION Script reserved words: Browser, Document, Form, FormURL, ION, or Mouse.

Variable Scope

ION Script documents can contain the ION_INCLUDE tag, which is used to insert the contents of another file (ION Script, HTML, or text) into the current ION Script document. Variables defined in the enclosing page have global scope, meaning they can be accessed from either the enclosing page or the included page. Variables

defined in included files have local scope, meaning they can be accessed only from the included page.

If a variable with the same name is declared in the enclosing page and the included page, the variable takes the value that was declared in the included page. For example, consider the following two ION Script files:

enclosing.ion:

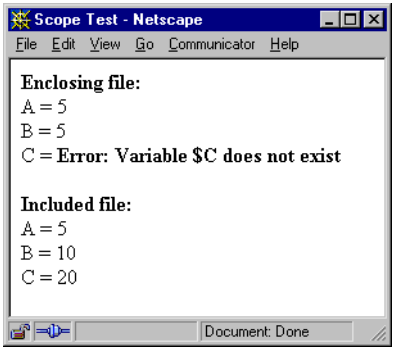
```
<ION_SCRIPT>
<ION_HEADER>
   <TITLE>Scope Test</TITLE>
   <VARIABLES>
      <VARIABLE_DECL NAME="A" VALUE="5" TYPE="INT"/>
      <VARIABLE_DECL NAME="B" VALUE="5" TYPE="INT"/>
   </VARIABLES>
</ION_HEADER>
<ION_BODY>
   <B>Enclosing file:</B><BR>
   A = <ION_VARIABLE NAME="$A"/><BR>
   B = <ION_VARIABLE NAME="$B"/><BR>
   C = <ION_VARIABLE NAME="$C"/>
   <ION_INCLUDE SRC="ion://included.ion"/>
</ION_BODY>
</ION_SCRIPT>
```

included.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="B" VALUE="10" TYPE="INT"/>
      <VARIABLE_DECL NAME="C" VALUE="20" TYPE="INT"/>
   </VARIABLES>
</ION_HEADER>
<ION_BODY>
   <BR>
   <B>Included file:</B><BR>
   A = <ION_VARIABLE NAME="$A"/><BR>
   B = <ION_VARIABLE NAME="$B"/><BR>
   C = <ION_VARIABLE NAME="$C"/>
</ION_BODY>
</ION_SCRIPT>
```

In this example, the enclosing page declares variables A and B, each set to 5. The included page also declares variable B, but sets the value to 10. The value of B in the included page becomes 10. Although variable A was not declared in the included page, we have access to its value from the enclosing page. The reverse is not true, however: the variable C, declared in the included page, is not accessible in the

enclosing page because its scope is limited to the included page. The following figure shows the result of loading `Enclosing.ion` in the browser:



Variable Persistence

Variables can be persistent across ION Script pages. Once a variable is declared as persistent, it is available to all ION Script pages that are called via an ION Script event or ION_LINK, without having to re-declare the variable in each page. If the variable is declared as not persistent (PERSIST set to FALSE or no PERSIST attribute), or not declared at all, the variable can be referenced only in the page in which it is declared or assigned a value, and in included pages. If you declare a persistent variable and pass it to a page that declares a variable (persistent or not) with the same name, the new declaration is not used—the value passed from the previous page is used.

System Variables

System variables are built-in ION Script variables used to hold useful information about ION Script applications. Each system variable can have multiple variables associated with it. You refer to these variables using the following general syntax:

```
$SystemVariable.VariableName
```

where `$SystemVariable` is $Browser, $Document, $Form, $FormURL, $ION, or $Mouse, and `VariableName` is the specific variable name for the specified system variable. Some system variables have only built in variables associated with them, such as $Mouse, which has $Mouse.x and $Mouse.y. Other system variables are assigned variables by the user, such as $Form, when the user defines an INPUT element.

Note

System variables are case-insensitive, except for any variables specified by the $Browser system variable. The $Browser system variable is used to specify actual CGI environment variables, such as HTTP_USER_AGENT. CGI environment variables are case-sensitive.

$Browser

The $Browser system variable can be used to access any CGI environment variable. CGI environment variables contain various information about the request, such as the server software being used, the hostname making the request, and whether the request was made using the GET or POST method.

One CGI environment variable in particular, HTTP_USER_AGENT, is useful in determining the type and version of the browser being used to request the page. Determining the browser version and type is useful when your application has content that cannot be rendered by all browsers. Browser vendors often incorporate proprietary “extensions” to HTML that add functionality to their product, but that are not supported by other browsers. It is generally recommended that you limit your use of browser-specific, proprietary features, but when used, you are strongly encouraged to provide alternate content that can be viewed by any browser. One technique for ensuring that users of different browsers will all see your content in the manner best suited to their browsers is to determine the browser type and version, and redirect users to the appropriate page for their browser. It is beyond the scope of this document to discuss these techniques in detail, but extensive information on the subject can be found in many HTML and JavaScript/VBScript references.

The CONTAINS operator can be used to check the $Browser.HTTP_USER_AGENT variable for a browser type or version. For example, the following code could be used to determine if the browser is Microsoft Internet Explorer:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IF EXPR="$Browser.HTTP_USER_AGENT CONTAINS 'MSIE'">
      You are using Internet Explorer.
   </ION_IF>
</ION_BODY>
</ION_SCRIPT>
```

See http://hoohoo.ncsa.uiuc.edu/cgi/env.html for a list of CGI environment variables.

Warning

Assigning `$Browser.REMOTE_USER` as a string variable to the VALUE attribute can be dangerous because it is frequently an undefined system variable. When VALUE is undefined, IDL reports an error. However, it is possible that code that uses `$Browser.REMOTE_USER` could be implemented and tested without error on one system and fail if ported to another system.

$Document

The $Document system variables store the metadata defined in the header section. For example, if the header section of your ION Script page contains the tag

```
<TITLE>My Title</TITLE>
```

then $Document.TITLE contains the text “My Title”.

APPLICATION

The application name, defined with the <APPLICATION> tag.

AUTHOR

The document author, defined with the <AUTHOR> tag.

COPYRIGHT

The copyright statement, defined with the <COPYRIGHT> tag. To add a copyright statement to your ION Script page, you can use the COPYRIGHT and ION_VARIABLE tags as follows:

```
<ION_SCRIPT>
<ION_HEADER>
   <COPYRIGHT>Copyright 2004 Research Systems, Inc.</COPYRIGHT>
</ION_HEADER>
```

```
<ION_BODY>
   <ION_VARIABLE NAME="$Document.COPYRIGHT"/>
</ION_BODY>
</ION_SCRIPT>
```

You can control whether or not the © symbol appears before the copyright statement using the Format tab of the configuration utility. See "Include © when displaying $Document.COPYRIGHT" on page 24.

DATE

The date the document was created, defined with the <DATE> tag.

EVENT

A variable of type STR that contains the name of the event that triggered the current page to be loaded. If no event triggered the page, its value is 'page'. Therefore, 'page' is an illegal name for an event. The following two ION Script pages demonstrate the use of $Document.EVENT.

page1.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="PASSEVENT" ACTION="page2.ion"/>
   </EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
      <ION_BUTTON EVENT="PASSEVENT" LABEL="Load page via an event"
         TYPE="BUTTON" /><BR>
   </ION_FORM>
   <FORM ACTION="http://myserver/cgi-bin/ion-p">
      <INPUT TYPE="HIDDEN" NAME="page" VALUE="ion://page2.ion">
      <INPUT TYPE="SUBMIT" VALUE="Load page via an HTML form">
   </FORM>
</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IF EXPR="$Document.EVENT NE 'page' ">
      The <ION_VARIABLE NAME="$Document.EVENT"/> loaded this page
   <ION_ELSE/>
      This page was not loaded from an event
   </ION_IF>
</ION_BODY>
```

```
</ION_SCRIPT>
```

LASTUPDATE

The date the document was last updated, defined with the <LASTUPDATE> tag.

TITLE

The document title, defined with the <TITLE> tag.

$Form

$Form variables are variables that arrived in the page as the result of one of the following:

- An HTML or ION Script form being submitted.
- An ION_LINK being clicked.
- A value passed to the page in a URL query string.

$Form variables are always treated as strings. Form variables are referred to using the following syntax:

$Form.*Formvar*

Formvar can be any of the following:

- The NAME attribute of an <INPUT> tag. For example, consider the following form field definition:

  ```
  <INPUT NAME="MyVar" TYPE="TEXT" VALUE="Hello">
  ```

  On the action page of this form, you would refer to the value of this form field as follows:

  ```
  $Form.MyVar
  ```

- The NAME attribute of an ION_RADIO button.
- The NAME attribute of an ION_CHECKBOX.
- The NAME*n* attribute of an ION_LINK.
- The name of any name/value pairs passed in a URL query string. See “Passing Name/Value Pairs in a URL” on page 75 for more information.

Note

You cannot access the value of a $Form variable on the page in which the form field is defined. The value is not available until the form has been submitted.

Using $Form Variables in Numeric Expressions

To use a $Form variable in a numeric expression (using ION_IF or ION_EVALUATE), you must declare a variable as an INT or DOUBLE using VARIABLE_DECL, and assign this variable the value $Form.*NAME*, where *NAME* is the NAME attribute of the form field or ION_LINK, or the name in a name/value pair passed in a URL. This converts the string variable — $Form variables are always string variables — to the type you desire. When writing the expression, you then use this new variable instead of using the $Form variable. For example:

```
<ION_FORM>
   <INPUT NAME="Age" TYPE="TEXT" VALUE="35">
</ION_FORM>
```

In the page that is called, you would define a variable as follows:

```
<VARIABLE_DECL NAME="AGE" VALUE="$Form.Age" TYPE="INT">
```

This variable needs to be defined only on pages that use the $Form variable in a numeric expression. You then use this new variable in place of the $Form variable in a numeric expression, such as in the following example:

```
<ION_IF EXPR="$AGE GT 30"> <!--$AGE is $Form.Age -->
```

The reason for this requirement is that $Form variables are always treated as strings. Since strings and numerics cannot be used together in a numeric expression (except to concatenate a numeric to a string using the + operator), creating a numeric variable equal to the form variable allows you to use the form variable in a numeric expression, such as an expression using a comparison operator.

$FormURL

The $FormURL system variable is equivalent to the $Form variable except that $FormURL returns the string in URL-encoded format. In a URL-encoded string, special characters such as carriage returns and tabs are replaced with codes representing that character. ION Script defines special characters as:

- ASCII values up to 32
- Any of the following: ; / ? : @ = & < > " # % { } | \ ^ ~ [ ] ` +
- ASCII values from 127-255

When URL encoded, these special characters are replaced by %*hh*, where *hh* is the hexadecimal ASCII value of the character. Carriage returns are replaced by %0D%0A. The IDL program `strdecode.pro` (located in the `ion_script/examples` directory) can be used to decode the URL-encoded string in an <IDL> block.

When Should You Use $FormURL?

$FormURL is useful in cases where the user might enter characters that would cause problems when passed in a string directly to IDL, such as carriage returns and quotation marks, as illustrated in the following example.

Example 1

In this example, a text field is used to enter the x-axis label for a plot. Assume the user enters a label containing an apostrophe (single quotation mark), such as “Doug's Data”. The value of $Form.LABEL is the string “Doug's Data”. If $Form.LABEL is enclosed in single quotation marks in an <IDL> block, such as with the IDL command

```
label = '$Form.LABEL'
```

a syntax error would occur because this would be interpreted by IDL as

```
label = 'Doug's Data'
```

This would cause the string to be terminated prematurely in IDL. The $FormURL.LABEL variable is the string “Doug%27s%20Data”. This can be decoded in an <IDL> block using the IDL program `strdecode.pro` as follows:

```
label = strdecode('$FormURL.LABEL')
```

Now, `label` is the string “Doug's Data”.

FormURL1.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="PLOT" ACTION="ion://FormURL2.ion"/>
   </EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
      X-axis Label: <INPUT TYPE="TEXT" NAME="xlabel">
      <BR>
      <ION_BUTTON TYPE="BUTTON" EVENT="PLOT" LABEL="Show Plot"/>
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

FormURL2.ion:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IMAGE TYPE="DIRECT">
      <IDL>
```

```
            label = strdecode('$FormURL.xlabel')
            plot, [0,1], xtitle=label
         </IDL>
      </ION_IMAGE>
   </ION_BODY>
   </ION_SCRIPT>
```

Example 2

Another case in which $FormURL is useful is if your ION Script form contains a textarea to allow the user to enter (or paste in) data that is delimited with commas, tabs, spaces, and/or carriage returns. The `strdecode.pro` program can convert such data into the format you require.

In this example, a textarea is used to enter a two-dimensional array of data for a contour plot. By setting the ARRAY keyword in the call to `strdecode`, we allow the user to enter a two-dimensional array, where the elements of each row can be separated by spaces, tabs, or commas, and each row is terminated with a carriage return. Therefore, if the user enters data containing multiple rows and columns, this application preserves the format of the data. Note that `strdecode` creates a string array, therefore, this array must be converted to a numeric array in this example.

textarea1.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="DRAW" ACTION="ion://textarea2.ion"/>
   </EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
     Enter contour data:
     <BR>
     <TEXTAREA NAME="data" ROWS="5"></TEXTAREA>
     <BR>
     <ION_BUTTON TYPE="BUTTON" EVENT="DRAW" LABEL="Draw Contour"/>
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

textarea2.ion:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IMAGE TYPE="DIRECT" BORDER="0">
      <IDL>
         ; Decode the URL-encoded string into a 2D array
         data = strdecode('$FormURL.data', /array)
```

```
            ; Convert to float array
            data = float(data)
            ; Smooth the data
            data = min_curve_surf(data)
            ; Draw the contour plot
            contour, data
         </IDL>
      </ION_IMAGE>
   </ION_BODY>
   </ION_SCRIPT>
```

Using strdecode.pro

The IDL program `strdecode.pro` is provided as an example of how you can decode URL-encoded strings. The `strdecode` command uses the following syntax:

Result = STRDECODE(*String* [, /ARRAY] [, /NO_SPLIT])

The *String* argument is the $FormURL string to be decoded.

If ARRAY is set, spaces, commas, tabs, and carriage returns are treated as delimiters, and the scalar string is converted into a string array. If NO_SPLIT is set, a one-dimensional (vector) string array will be returned, otherwise the result will be a two-dimensional string array.

Note

For a two-dimensional array, the first input row is used to determine the number of columns. Rows that contain fewer columns will be filled with null strings, while extra columns will be ignored.

If NO_SPLIT is set, carriage returns will not be used to split the string into a string array. Carriage returns will remain within the string as characters with the value STRING(13b).

Assume the following data is entered into a textarea (<CR> is a carriage return):

```
1,2<CR>
3,4<CR>
5,6<CR>
```

The following table illustrates how the different keyword combinations effect the value returned by `strdecode`. returns the following *Result* when decoding $FormURL.DATA using the various keyword combinations:

Keywords Set	HELP, Result	PRINT, Result
None	String = Array [3]	1,2 3,4 5,6
/ARRAY	String = Array [2, 3]	1 2 3 4 5 6
/NO_SPLIT	String	'1,2 3,4 5,6'
/ARRAY, /NO_SPLIT	String = Array [6]	1 2 3 4 5 6

Table 3-1: Effect of different keyword combinations with strdecode.pro

$ION

The $ION system variables are read-only values that contain information about ION Script.

IDLURL

$ION.IDLURL is a read-only string value specifying the URL used to call the ION Script image engine, `ion-i.exe`. This system variable is used as the value of the DATA attribute of the ION_OBJECT tag (when OBJTYPE is OBJECT), or the SRC attribute of the ION_OBJECT tag (when OBJTYPE is EMBED). Refer to “ION_OBJECT” on page 173 for examples on using $ION.IDLURL.

temp

$ION.temp is a read-only string value specifying the location in which ION Script writes temporary files before transferring them to the client. This location is specified on the “Files” tab of the ION Script Configuration Utility (Windows), or in the “Files” section of the `.ionsrc` file (UNIX). $ION.temp can be used to write files to the temporary directory when writing files from your ION Script application.

uniqueID

$ION.uniqueID is a read-only integer value representing the process number of the ion-p process. Because the process number is unique each time ion-p is executed, this number can be used to create unique filenames when writing files from IDL. This prevents files written by your ION Script application from being overwritten when multiple users are running your application. For an example in which $ION.temp and $ION.uniqueID are used to generate a filename, see the example application `largeData.ion` in the `examples` directory.

Version

$ION.version is a read-only string value specifying the version of ION Script you are currently using.

$Mouse

The $Mouse system variables store the x and y coordinates of the mouse cursor when the mouse is clicked on an ION_IMAGE, an ION_BUTTON of TYPE="IMAGE", or any HTML image created with <INPUT TYPE="IMAGE">.

x

$Mouse.x stores the x location of the cursor.

y

$Mouse.y stores the y location of the cursor. Note that when you click an image, the browser considers the origin (0, 0) to be the upper left corner of the image. In IDL, however, the origin of the image is the lower left corner. Therefore, when you want to pass the y location of the mouse click to IDL, you must subtract $Mouse.y from the height of the image. For example, to annotate a 300 x 300 image at the location of the mouse click using the XYOUTS procedure, you could use IDL code similar to the following:

```
XYOUTS, $Mouse.x, 300-$Mouse.y, '. You clicked here', /DEVICE
```

To see this behavior in an ION Script application, run the example `mouse.ion` in the `ION61/ion_script/examples` subdirectory of your ION Script installation. See “ION_IMAGE” on page 159 for another example using $Mouse.

The default values of $Mouse.x and $Mouse.y are 0.0, therefore, you can check to see if the user has clicked the mouse with the statement

```
<ION_IF EXPR="$Mouse.x NE 0.0">
```

See “Using the $Mouse System Variable” on page 107 for an example that employs this technique.

Expressions

An expression is a combination of values, variables, and/or other expressions separated by operators, or simply a single value or variable.

The following are examples of expressions:

'Friday'	1 + 3
$A + $B	4.0
$E = $M * $C^2	$String1 EQ $String2
$Browser.type	$Variable = 4 + 3
$Form.name	$X = $A + $B
$Variable = (34 GE 341)	$LastName = 'Smith'

Values

Values used in expressions can be any of the following:

- Variables of the form $*varname*
- String literals of the form 'string text'
- Numbers of the form x or x.x
- Numbers using scientific notation of the form *x*E*y*, *x*e*y*, *x*D*y*, or *x*d*y*, where *x* is any real number and *y* is a positive or negative exponent. For example, 2E6 is the same as $2 * 10^6$. Note that in ION Script there is no difference between the characters e, E, d, and D.
- Booleans of the form TRUE, FALSE, true, or false. For example, $A EQ TRUE

Expression Types

There are three main types of expressions. They are:

- Numeric expressions
- Boolean expressions
- String expressions

Numeric Expressions

Numeric expressions are expressions that evaluate to a number. The following are examples of numeric expressions:

- 1 + 3
- 4.0
- $a + $b (assuming a and b are both numbers)
- $Mouse.x

Numbers can be one of two types: DOUBLE (double precision, such as 1.2345) or INT (integer, such as 1).

All numeric expressions are evaluated as double-precision values, regardless of whether the variables or values in the expression are integers or double-precision values. For example, if you declare variables A and B as integers, the result of $A+$B will be a double. If, however, you assign the result of $A+$B to a variable that was declared as an integer, then the result will be an integer.

Note

ION Script uses the C printf conversion specifier %g with 16 significant digits when displaying double-precision values. If, when converted to scientific notation, the value's exponent is less than −4 or greater than 16, the value will be displayed using scientific notation. Otherwise, the value is printed as a floating-point value. Trailing zeros in the fractional part of the value are not printed.

Boolean Expressions

Boolean expressions are expressions that return a boolean value, that is, true or false. The values used in the expression are numbers or strings. The operators that return boolean values are:

GT, GE, LT, LE, EQ, NE, ISTYPE, AND, OR, NOT

The following are examples of boolean expressions:

- $Temp GE 212
- $X EQ $Y
- $Mouse.x GT 128
- $Form.UserEntry EQ 'Form1'
- $Name ISTYPE 'STR'

- $BOOL1 AND $BOOL2

If you declare a variable as a BOOL, and then later reassign it a numeric value, the variable is assigned a value of FALSE if the expression evaluates to zero, and TRUE otherwise. For example, if $C was declared as type BOOL, and later reassigned the value $C = $A + $B, then $C would be assigned the value FALSE if $A + $B evaluates to 0, and TRUE if $A + $B evaluates to anything other than 0.

String Expressions

String expressions are expressions that return strings. Strings can be used with five operators: EQ, NE, +, CONTAINS, and ISTYPE. The + operator is used to *concatenate*, or combine, two strings. For example, the expression 'Hello' + ' World!' yields 'Hello World!'.

The following are examples of string expressions:

- 'Your name is ' + $Name
- 'RSI'
- $Form.*variable* (All $Form variables are treated as strings. See “INPUT” on page 129 for information on when you should declare variables corresponding to $Form variables.)
- 5 + 'gallons'. When adding a numeric value to a string value, the result of the numeric value is concatenated as a string to the string value.
- $Name CONTAINS 'Smith'

Note that $Name in the above examples must be a string.

If you declare a variable as type string and assign it a numeric value, the number will be converted to double-precision and then to a string. For example, the following code creates a string variable and assigns the value 35.4:

```
<VARIABLE_DECL NAME="A" VALUE="35.4" TYPE="STR"/>
```

If you write the value of this variable to the page, it will read as 35.40 instead of 35.4. To prevent the number from being converted to a double-precision value, enclose it in single quotation marks as follows:

```
<VARIABLE_DECL NAME="A" VALUE=" '35.4' " TYPE="STR"/>
```

Operators

Operators are used to perform comparisons and mathematical operations on the terms of an expression. Operators are used between values or expressions in the general form *value1* OP *value2*. Some operators only accept values of a certain type. If the value for a variable is not a type supported by the operation, the operation will fail.

Mathematical Operators

ION Script's mathematical operators are listed in the following table.

Operator	Description	Types Accepted
+	Addition: Add *value1* to *value2*.	variables, numbers
-	Subtraction. Subtract *value2* from *value1*	variables, numbers
/	Division. Divide *value1* by *value2*	variables, numbers
*	Multiplication. Multiply *value1* by *value2*	variables, numbers
MOD	Modulus. Returns the integer remainder of dividing *value1* by *value2*	variables, numbers
^	Raise to the Power Of. Returns *value1* raised to the power *value2*	variables, numbers

String Operators

ION Script's string operators are listed in the following table.

Operator	Description	Types Accepted
+	String concatenation. 'String1' + 'String2' creates 'String1String2'.	variables, numbers, strings
CONTAINS	String contains. *value1* CONTAINS *value2* returns true if *value2* is found in *value1*, returns false otherwise.	variables, strings

Comparison Operators

ION Script's comparison operators are listed in the following table.

Operator	Description	Types Accepted
GT	Greater Than. TRUE if *value1* is greater than *value2*	variables, numbers
GE	Greater Than or Equal To. TRUE if *value1* is greater than or equal to *value2*	variables, numbers
LT	Less Than. TRUE if *value1* is less than *value2*	variables, numbers
LE	Less Than or Equal To. TRUE if *value1* is less than or equal to *value2*	variables, numbers
EQ	Equality. TRUE if *value1* is equal to *value2*	variables, numbers, strings
NE	Not Equal To. TRUE if *value1* is NOT equal to *value2*	variables, numbers, strings
ISTYPE	TRUE if the data type of the variable specified by *value1* corresponds to *value2*, a string representing the variable type. *value2* can be one of the following string literals: 'BOOL', 'STR', 'INT', 'DOUBLE', or 'UNDEF'.	variables, numbers, strings

Logical Operators

ION Script's logical operators are listed in the following table. Note that values used with logical operators can be values, variables, or expressions, but they must evaluate to a boolean value.

Operator	Description	Types Accepted
AND	Logical AND. *value1* AND *value2* returns true if both *value1* and *value2* evaluate to true. Returns false otherwise.	Boolean value or variable, or expression that evaluates to a boolean
OR	Logical OR. *value1* OR *value2* returns true if either *value1* or *value2* evaluates to true. Returns false otherwise.	Boolean value or variable, or expression that evaluates to a boolean
NOT	Logical negation. NOT *value* returns true if *value* is false and returns false if *value* is true.	Boolean value or variable, or expression that evaluates to a boolean

The following are examples of valid expressions using the logical operators. For these examples, assume

```
$BOOL1 = true
$BOOL2 = false
$A = 1
$B = 2
```

Type of operand	Expression	Result
Boolean variables	$BOOL1 AND $BOOL2	false
	$BOOL1 OR $BOOL2	true
Boolean values	true AND false	false
	true OR false	true
Expressions that evaluate to booleans	($A NE $B) AND ($A EQ 1)	true
	($A EQ 5) OR ($B EQ 5)	false
	NOT ($A EQ $B)	true
	$BOOL1 AND ($A EQ 1)	true

Operator Precedence

The following table lists the ION Script operators in order of precedence. All operators are left associative, meaning that operators of equal precedence (such as + and -) are evaluated from left to right.

Precedence	Operators
Highest	( )
↓	- (negation)
↓	^
↓	NOT
↓	*, /, MOD
↓	+, -
↓	GT, GE, LT, LE, CONTAINS, ISTYPE
↓	AND
Lowest	OR
	EQ, NE

Grouping Expressions With Parentheses

Parentheses are used in ION Script to control the order of evaluation of expressions. Expressions that are contained inside parentheses are evaluated first by ION Script. When multiple sets of parentheses are nested within one another, the expression contained in the innermost set of parentheses is evaluated first.

For example, consider the following code:

```
<ION_IF EXPR="$Month EQ 'April' OR $Month EQ 'May'
   AND $Temp EQ 70 ">
```

This statement is equivalent to the following:

```
<ION_IF EXPR="$Month EQ 'April' OR
   ($Month EQ 'May' AND $Temp EQ 70)">
```

If your intent was to require $Temp to be 70, but allow the $Month to be April or May, you would need to override the precedence as follows:

```
<ION_IF EXPR="($Month EQ 'April' OR $Month EQ 'May') AND
   $Temp EQ 70 ">
```

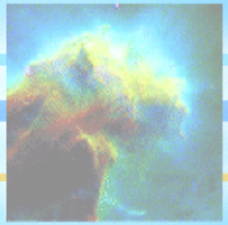
Tip

It is good programming practice to use parentheses to control the order of evaluation, even when not technically necessary. This ensures that your expressions are being evaluated in the order you intended, and makes your code more readable.

Examples

The following table gives several examples of the order in which expressions are evaluated in ION Script:

Expression	Result
10 * 5 - 2	48
10 * (5 - 2)	30
3 + 4 * 2 ^ 2 / 2	11
(3 + (4 * 2) ^ 2 / 2)	35



Chapter 4: Creating ION Script Applications

The following topics are covered in this chapter:

- Specifying URLs
- HTML Forms vs. ION Script Forms
- Passing Name/Value Pairs in a URL
- Handling Multiple Selections in a SELECT Element
- Passing IDL Variables to ION Script
- Using Frames with ION Script
- Using JavaScript and VBScript
- Graphics in ION Script
- Using Special and International Characters
- Example: Creating a Complete Application

Specifying URLs

The following ION Script tags have an attribute that takes a URL as its value:

ION Script Tag	Attribute	Allowable Protocols
<EVENT_DECL>	ACTION	http, https, file, ion
<ION_BODY>	BACKGROUND	http, https, file
<ION_BUTTON TYPE="IMAGE">	SRC	http, https, file
<ION_IMAGE>	SERVER	http, https
<ION_INCLUDE>	SRC	http, https, file, ion

In addition, the following HTML tags have an attribute that can take a URL that points to an ION Script page:

HTML Tag	Attribute	Allowable Protocols
<FORM>	ACTION	http, https, file, ion
<FRAME>	SRC	http, https, file, ion
<IFRAME>	SRC	http, https, file, ion

The general syntax for URLs is as follows:

protocol://[*username*:*password*@][*hostname*][:*port*]/*path*

protocol must be one of the following:

- **http -** used to specify the location of a file on a remote network such as the WWW.
- **https -** used to specify the location of a file on a remote network handling secure transactions.

- **file** - used to specify a file (except for `.ion`) that is accessible on a particular host computer.
- **ion** - used to specify the URL of a `.ion` file, either as an event action or an included file. Files specified using the ion:// protocol are parsed by the ION Script parser. URLs using the ion:// protocol cannot be used in HTML tags, nor can they be entered in the browser's location field.

Note

Because UNIX is case-sensitive, make sure if using a UNIX Web server that URLs used in your ION Script and HTML pages use the same case as the files to which they refer.

HTTP URLs

HTTP (Hypertext Transfer Protocol) is the protocol used to transfer HTML files over a network. Cases in which you would use the http:// protocol in your ION Script applications include:

- when specifying an HTML page as the EVENT attribute to the ION_BUTTON or ION_IMAGE tag
- when including an HTML page using the ION_INCLUDE tag
- when adding a link to another HTML page from your ION Script page, using the HTML <A> tag or the ION_LINK tag

HTTPS URLs

HTTPS (Hypertext Transfer Protocol, Secure) is the protocol used to transfer files over a network using secure transactions. The https:// protocol behaves like the http:// protocol, but is intended to provide secure communication of information such as credit-card numbers.

You need only specify this protocol to take advantage of this security, which the Web server and browser handle for you. If you want to use https://, you can change existing references to http:// in the following three places:

- The Image Server URL configuration setting (used to find `ion-i`)
- The ION Script Parser URL configuration setting (used to find the parser via a URL) — the ion:// protocol uses this setting
- The `.ion` and `.html` code used with ION

File URLs

File URLs are used to access files on a particular host computer, as opposed to files located on a remote network such as the WWW. The file:// protocol is intended to provide access to text and other files on your Web server, such as when you want to include a file in your ION Script page using the ION_INCLUDE tag.

The syntax for URLs that use the file:// protocol is as follows:

file:///*path*/*filename*

If the path is absolute, the leading / must be present, as in the following examples:

UNIX: file:////usr/local/etc/readme.txt

Windows: file:///c|/projects/ionscript/readme.txt

Note

For Windows drives, a vertical line is used in place of the colon when specifying the drive.

ION URLs

The ion:// protocol causes the page referred to in the URL to be parsed by the ION Script parser. When the ion:// protocol is used, the page referred to in the URL will be parsed as an ION Script file no matter what extension the file uses. For example, the following EVENT definition causes the file `myfile.xxx` to be parsed as an ION Script file:

```
<EVENT_DECL NAME="event" ACTION="ion:///c:\ion\myfile.xxx"/>
```

The only times you specify the ion:// protocol are when you define an EVENT attribute as another ION Script page, and when you include an ION Script file using the SRC attribute of ION_INCLUDE. Note that you cannot use the ion:// protocol in the following ways:

- For the SRC attribute of ION_BUTTON, which must point to an existing image file
- In HTML tags
- In the location field of the browser

Absolute Paths vs. the ION Search Path

How you specify file:// and ion:// URLs determines whether an absolute path or the ION Search Path is used to locate the file. The ION Search Path is a search path

specified in the configuration utility or `.ionsrc` file (see "ION Search Path" on page 20).

Absolute paths to ion:// and file:// URLs must start with /. For example:

```
ion:///f:/projects/ionscript/include.ion
```

If only a filename is entered, the ION Search Path is used for the path and the leading / may be omitted, For example, if the following URL is used, ION Script will search for `include.ion` in the ION Search Path:

```
ion://include.ion
```

If only a filename is entered, but you do include the leading /, the absolute search fails and the ION Search Path is used. For example:

```
ion:///include.ion
```

If the entire path is used, and you do not include the leading /, as in the following example, the URL is NOT valid:

```
ion://f:/projects/ionscript/include.ion
```

HTML Forms vs. ION Script Forms

ION Script forms, created with the ION_FORM tag, are beefed-up versions of the HTML form. An HTML form is limited to one action (defined by the ACTION attribute). An ION Script form can have a different EVENT for each ION_BUTTON, ION_IMAGE, and ION_LINK contained in the form.

In addition, ION Script forms can contain the ION_CHECKBOX and ION_RADIO elements, which, unlike their HTML counterparts, maintain state when the page is reloaded.

If, however, you have an existing HTML form, you can still access the form values that are submitted without creating an entirely new ION Script form to replace the HTML version. Following are two ION Script pages—one uses an ION Script form, the other uses an HTML form. `page2.ion` is the action of each form. Load both `ionform.ion` and `htmlform.ion`, press the button, and each form passes the $Form variable to `page2.ion`:

`ionform.ion:`

```
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="PAGE2" ACTION="ion://page2.ion"/>
   </EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
      <INPUT NAME="Text1" TYPE="TEXT" VALUE="Hello">
      <ION_BUTTON EVENT="PAGE2" TYPE="BUTTON"/>
   </ION_FORM>
</ION_BODY>
```

`htmlform.ion:`

```
<ION_BODY>
   <FORM ACTION="http://myserver/cgi-bin/ion-p">
      <INPUT TYPE="HIDDEN" NAME="page" VALUE="page2.ion">
      <INPUT NAME="Text1" TYPE="TEXT" VALUE="Hello">
      <INPUT TYPE="SUBMIT">
   </FORM>
</ION_BODY>
```

`page2.ion:`

```
<ION_BODY>
<ION_VARIABLE NAME="$Form.Text1"/>
</ION_BODY>
```

Passing Name/Value Pairs in a URL

Any name/value pairs passed in a URL query string to the ION Script parser will become ION Script variables in the page loaded by the URL. For example, consider the URL in the following ION Script page:

page1.ion:

```
<ION_SCRIPT>
<ION_BODY>
   <A HREF="http://host/cgi-bin/ion-
p.exe?page=page2.ion&var1=5&var2=abc">Pass var1 and var2 to
page2.ion</A>
</ION_BODY>
</ION_SCRIPT>
```

When you click the link in the above page, the values var1 and var2 are passed to `page2.ion`. You can access these values on `page2.ion` as $Form.var1 and $Form.var2, as shown in the following example:

page2.ion:

```
<ION_SCRIPT>
<ION_BODY>
The value of var1 is <ION_VARIABLE NAME="$Form.var1"/>.
<BR>
The value of var2 is <ION_VARIABLE NAME="$Form.var2"/>.
</ION_BODY>
</ION_SCRIPT>
```

When you click the link on `page1.ion`, `page2.ion` displays the following:

```
The value of var1 is 5.
The value of var2 is abc.
```

Passing ION Script Variables in a URL

Instead of simply passing hard-coded values in a URL as in the above example, it is also possible to pass ION Script variables in a URL. To do so, you must insert the ION Script variable in the URL query string by using the ION_VARIABLE tag as follows:

page1.ion:

```
<ION_SCRIPT>

<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="A" TYPE="INT" VALUE="10"/>
```

```
   </VARIABLES>
</ION_HEADER>

<ION_BODY>

<A HREF=
http://host/cgi-bin/ion-p.exe?page=page2.ion&var1=
<ION_VARIABLE NAME="$A"/>>Page 2</A>

</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>

<ION_BODY>
   The value of the ION Script variable passed in the URL for this
page is <ION_VARIABLE NAME="$Form.var1"/>.
</ION_BODY>

</ION_SCRIPT>
```

When you click the link on `page1.ion`, `page2.ion` displays the following:

```
The value of the ION Script variable passed in the URL for this
page is 10.
```

Passing Multiple Values With the Same Name

If you pass a URL to the ION Script parser, and the URL contains a query string in which more than one name/value pair uses the same name, then a single $Form variable is created in which the values assigned to the variable are separated by the "|" character. For example, if you pass to the ION Script parser the URL

```
http://host/cgi-bin/ion-p.exe?var1=One&var1=Two&var1=Three
```

then a $Form variable named $Form.var1 will be created with the value "One|Two|Three".

This functionality is helpful when you use the MULTIPLE attribute to the SELECT element to allow the user to select multiple options from a drop-down list. In this case, each of the options selected by the user will be appended to the $Form variable and separated by the "|" character when the form is submitted. You must then parse the $Form variable to extract the values. For an example of how to use the MULTIPLE attribute of the SELECT tag, see "Handling Multiple Selections in a SELECT Element" on page 77.

Handling Multiple Selections in a SELECT Element

When the MULTIPLE attribute is specified for the HTML <SELECT> tag, the user is allowed to select multiple options. When a user selects multiple options in a SELECT element that resides in an ION_FORM, the $Form variable created on the page that is loaded when the form is submitted contains the value of all selected options, separated by the "|" character. For example, suppose an ION_FORM contains the following SELECT element:

```
<SELECT NAME="region" MULTIPLE>
    <OPTION>East</OPTION>
    <OPTION>West</OPTION>
</SELECT>
```

If the user selects both options and submits the form, the value of the $Form.region variable on the page that is loaded will be the string "East|West". You must then parse the value of $Form.region to extract the values contained in the string. The following sections illustrate two different methods of handling SELECT elements when the MULTIPLE attribute is used.

Parsing the $Form Variable in an <IDL> Block

This example demonstrates how you could use the IDL STRSPLIT function to extract multiple values from a $Form variable.

page1.ion:

```
<ION_SCRIPT>

<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="DISPLAY"  ACTION="ion:///page2.ion"/>
   </EVENTS>
</ION_HEADER>

<ION_BODY>

<ION_FORM>
   <SELECT NAME="Region" MULTIPLE>
      <OPTION>East</OPTION>
      <OPTION SELECTED>West</OPTION>
   </SELECT>
   <ION_BUTTON TYPE="BUTTON" EVENT="DISPLAY" LABEL="Display"/>
</ION_FORM>
```

```
</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>

<ION_BODY>

<ION_DATA_OUT>
   <IDL>
      str = '<ION_VARIABLE NAME="$Form.Region"/>'
      str_array = STRSPLIT(str, '|',/EXTRACT)
      parse_str_array, str_array
   </IDL>
</ION_DATA_OUT>

</ION_BODY>
</ION_SCRIPT>
```

parse_str_array.pro:

```
pro parse_str_array, array
   length = n_elements(array)
   for i=0,length-1 do begin
      print, array[i]
   endfor
end
```

Instead of printing the values contained in the $Form variable, you could pass the IDL variables to ION Script variables. Refer to the example “Passing IDL Variables to ION Script Example” on the ION Script Advanced Examples page (index_examples.ion).

Parsing the $Form Variable in a <SCRIPT> Block

This example demonstrates the use of a <SCRIPT> block to extract multiple values from a $Form variable.

page1.ion:

```
<ION_SCRIPT>

<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="DISPLAY"  ACTION="ion:///page2.ion"/>
   </EVENTS>
</ION_HEADER>
```

```
<ION_BODY>

<ION_FORM>
   <SELECT NAME="Region" MULTIPLE>
      <OPTION>East</OPTION>
      <OPTION SELECTED>West</OPTION>
   </SELECT>
   <ION_BUTTON TYPE="BUTTON" EVENT="DISPLAY" LABEL="Display"/>
</ION_FORM>

</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>
<ION_BODY>

<SCRIPT LANGUAGE="JScript">
var str = '<ION_VARIABLE NAME="$Form.Region" DISPLAY="FALSE"/>';
var str_array = str.split('|');
for (i=0;i<str_array.length;++i) {
   document.write(str_array[i]);
   document.write('<br>');
}
</SCRIPT>

</ION_BODY>
</ION_SCRIPT>
```

Passing IDL Variables to ION Script

There may be cases in which you need to access the value of an IDL variable in your ION Script application. This can be accomplished by writing IDL code that writes an ION_EVALUATE tag to a temporary file, and using ION_INCLUDE to include the IDL-generated ION_EVALUTE tag in your ION Script page.

For example, suppose you want to conditionally execute ION Script code depending on the value of an IDL variable. In the following example, the user enters a number and submits the form. The number entered is passed to IDL, which evaluates the square root of the number, and writes an ION_EVALUATE tag containing the square root to a temporary file. The ION Script page contains an ION_IF block that conditionally executes code depending on the value of the variable defined in the IDL-generated ION_EVALUTE tag.

First, we create the form in which the user enters a number:

page1.ion

```
<ION_SCRIPT>
<ION_HEADER>
  <EVENTS>
    <EVENT_DECL NAME="go" ACTION="page2.ion"/>
  </EVENTS>
</ION_HEADER>

<ION_BODY>

<ION_FORM NAME="form1">
   <INPUT NAME="num" TYPE="TEXT">
   <ION_BUTTON TYPE="BUTTON" EVENT="go" LABEL="Find square root"/>
</ION_FORM>

</ION_BODY>
</ION_SCRIPT>
```

In the action page of this form, page2.ion, we initialize two variables: the $TEMPFILE variable is used to hold the name of the temporary file that IDL will create, and the $IDLVALUE variable holds the value of the IDL variable.

Then we use ION_EVALUATE to define the value of the $TEMPFILE variable using the $ION.uniqueID system variable to ensure the name of our temporary file does not collide with any other filenames.

Next, we use ION_DATA_OUT to call an IDL function defined in writefile.pro. This IDL function accepts the value of $TEMPFILE and the value entered by the user,

calculates the square root, and defines the value of $IDLVALUE by writing the result as an ION_EVALUTE string to the file defined by $TEMPFILE.

The ION Script page then includes this temporary file, thereby re-evaluating the value of the $IDLVALUE variable. An ION_IF block then conditionally evaluates the $IDLVALUE variable written by IDL.

page2.ion

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="TEMPFILE" VALUE="''" TYPE="STR"/>
      <VARIABLE_DECL NAME="IDLVALUE" VALUE="0.0" TYPE="DOUBLE"/>
   </VARIABLES>
</ION_HEADER>
<ION_BODY>

<ION_EVALUATE EXPR="$TEMPFILE = $ION.temp + 'val' + $ION.uniqueID
   + '.ion'"/>
<ION_DATA_OUT DEBUG="FALSE">
   <IDL>
      writefile,'$TEMPFILE', $Form.num
   </IDL>
</ION_DATA_OUT>

<ION_INCLUDE SRC="ion:///$TEMPFILE" PRE="FALSE"/>

<ION_IF EXPR="$IDLVALUE GT 4">
   IDL returned a value greater than 4
<ION_ELSE/>
   IDL returned a value less or equal to 4
</ION_IF>

</ION_BODY>
</ION_SCRIPT>
```

writefile.pro

```
PRO writefile,tempfile, val
   A = SQRT(val)
   OPENW,lun,tempfile,/GET_LUN, ERROR=err
   if err NE 0 then begin
      print,!ERROR_STATE.MSG
      RETURN
   endif
   PRINTF,lun,'<ION_EVALUATE
   EXPR="$IDLVALUE='+STRCOMPRESS(A,/REMOVE_ALL)+'"/>'
   FREE_LUN,lun
END
```

Using Frames with ION Script

Frames are a useful tool for simplifying the navigation of a Web site, and offering different ways to present your information. It is often desirable to simultaneously display information from different sources in the same browser window, rather than force the user to navigate back and forth between pages. Creating framed ION Script pages is no different than creating framed HTML pages: you just need to know how to specify the frame source files, and how to target different frames. This section will show you how to create ION Script applications that use fixed framesets and floating frames.

Overview of Fixed Framesets

When you load a Web page that is divided into frames, you are loading an HTML frameset, defined with the <FRAMESET> tag. The <FRAMESET> tag divides the Web page either horizontally or vertically into multiple frames, each defined with its own <FRAME> tag. Each <FRAME> tag points to a Web page with the SRC attribute. For example, to create a page divided horizontally into two frames, we would create three HTML files similar to the following:

main.html:

```
<HTML>
<FRAMESET ROWS="25%, *">
   <FRAME NAME="Frame1" SRC="http://myserver/frame1.html">
   <FRAME NAME="Frame2" SRC="http://myserver/frame2.html">
</FRAMESET>
</HTML>
```

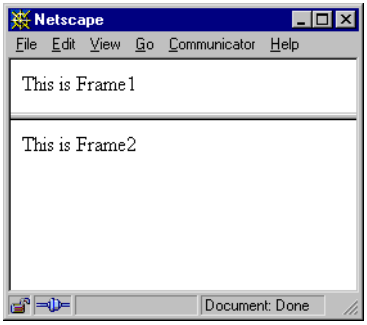
frame1.html:

```
<HTML>
This is Frame1
</HTML>
```

frame2.html:

```
<HTML>
This is Frame2
</HTML>
```

If you load `main.html` in your browser, you get the Web page shown in the following figure:

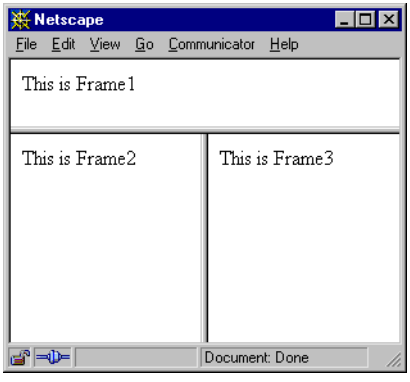


Note that `main.html` does not use the <BODY> tag; the <FRAMESET> tag is used instead.

We can further divide one of the frames in the previous example vertically into two frames by nesting a <FRAMESET> within the main <FRAMESET> as follows:

```
<HTML>
<FRAMESET ROWS="25%, *">
   <FRAME NAME="Frame1" SRC="http://myserver/frame1.html">
   <FRAMESET COLS="50%,*">
      <FRAME NAME="Frame2" SRC="http://myserver/frame2.html">
      <FRAME NAME="Frame3" SRC="http://myserver/frame3.html">
   </FRAMESET>
</FRAMESET>
</HTML>
```

This creates the following Web page:



Tip

Note the way that we referred to the HTML files in the SRC attribute of the <FRAME> tag. For example, consider the following HTML:

```
<FRAME NAME="Frame1" SRC="http://myserver/frame1.html">
```

In this case, the file `frame1.html` must be located in your Web server's default document directory, such as `htdocs`. To allow us to keep the HTML files in the same location as the ION Script files, we can change the SRC attributes of our <FRAME> tags to load the HTML page using `ion-p` instead:

```
<FRAME NAME="Frame1" SRC="ion-p?page=frame1.html">
```

We can also keep the page `main.html` in the same directory as the `.ion` files, and load the page using the following URL format:

`http://`*`servername`*`/cgi-bin/ion-p?page=main.html`

Loading ION Script Pages in Frames

In the above examples, the source of all our frames were HTML files. Frames can also contain ION Script pages. For example, we'll create a frameset with two frames, where the upper frame contains an ION Script page, and the lower frame initially contains a blank HTML page. When the user clicks the button in the upper frame, the ION Script form is submitted, and the target of that form is the lower frame. The ION_FORM tag uses the TARGET attribute to define the window in which the EVENT page is loaded upon form submission.

`main.html:`

```
<HTML>
<FRAMESET ROWS="20%,*">
   <FRAME NAME="upper" SRC="ion-p?page=frame1.ion">
   <FRAME NAME="lower" SRC="ion-p?page=blank.html">
</FRAMESET>
</HTML>
```

`frame1.ion:`

```
<ION_SCRIPT>
<ION_HEADER>
  <EVENTS>
    <EVENT_DECL NAME="PLOT" ACTION="ion://frame2.ion"/>
  </EVENTS>
</ION_HEADER>
```

```
<ION_BODY BGCOLOR="#ADD8E6">
   <ION_FORM TARGET="lower">
    <B>Display Value</B> <INPUT NAME="DISP_VALUE" TYPE="TEXT"
         VALUE="30" SIZE="3"/>
    <ION_BUTTON TYPE="BUTTON" LABEL="Show Plot" EVENT="PLOT" />
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

frame2.ion:

```
<ION_SCRIPT>
<ION_HEADER>
  <VARIABLES>
    <VARIABLE_DECL NAME="DISP_VALUE" VALUE="$Form.DISP_VALUE"
     TYPE="INT" />
  </VARIABLES>
</ION_HEADER>
<ION_BODY>
   <ION_IF EXPR="$DISP_VALUE LE 0">
      <tt> <ION_VARIABLE NAME="$DISP_VALUE" /> </tt> is too small!
   <ION_ELSE/>
      <ION_IMAGE TYPE="DIRECT">
         <IDL>
            loadct, 5
            show3, dist($DISP_VALUE)
         </IDL>
      </ION_IMAGE>
   </ION_IF>
</ION_BODY>
</ION_SCRIPT>
```

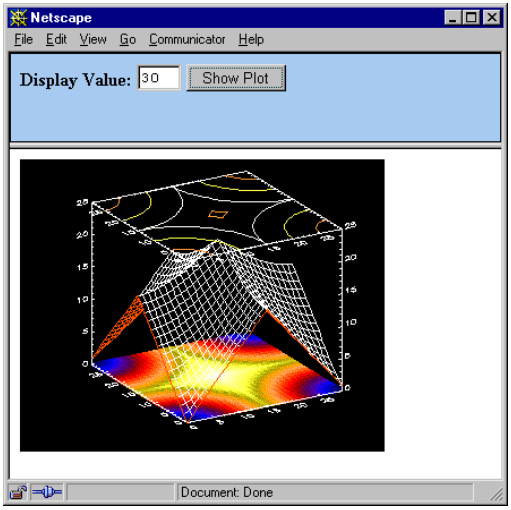
blank.html:

```
<HTML>
</HTML>
```

Load `main.html` in your browser using a URL such as

```
http://myserver/cgi-bin/ion-p?page=main.html
```

When you click the “Show Plot” button, you’ll see the page shown in the following figure:



Note that when loading an ION Script page through an HTML tag such as <FRAME>, you must explicitly pass name/value pairs in the URL for any variables required by the ION Script page. For example, suppose in the previous example that you want to pass the initial value displayed in the text field to `frame1.ion` instead of hard-coding the value. The page `frame1.ion` will contain a reference to the variable $Form.INITIAL:

```
<ION_SCRIPT>
<ION_HEADER>
  <EVENTS>
    <EVENT_DECL NAME="PLOT" ACTION="ion://frame2.ion"/>
</EVENTS>
</ION_HEADER>
<ION_BODY BGCOLOR="#ADD8E6">
   <ION_FORM TARGET="lower">
    <B>Display Value</B> <INPUT NAME="DISP_VALUE" TYPE="TEXT"
         VALUE="<ION_VARIABLE NAME="$Form.INITIAL"/>" SIZE="3"/>
    <ION_BUTTON TYPE="BUTTON" LABEL="Show Plot" EVENT="PLOT" />
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

In order for $Form.INITIAL to be defined when `frame1.ion` is executed, we must pass the INITIAL parameter in the URL that loads `frame1.ion`. Therefore, `main.html` would need to be changed as follows:

```
<HTML>
<FRAMESET ROWS="20%,*">
   <FRAME NAME="upper" SRC="ion-p?page=frame1.ion&INITIAL=30">
   <FRAME NAME="lower" SRC="ion-p?page=blank.html">
</FRAMESET>
</HTML>
```

Floating Frames

Floating frames are a feature introduced in Microsoft Internet Explorer 3.0. Unlike a fixed frameset, in which the frames are always anchored to the edge of the browser or to another frame, floating frames can exist as a separate island inside a Web page. Floating frames are created with the <IFRAME> tag, which exists inside the <BODY>. Therefore, floating frames can be inserted anywhere in the HTML stream, just like images, tables, and other HTML elements, and can scroll entirely out of the browser. This can be useful if you have information that applies to a specific section of your page, but you don't want that information to remain on screen at all times.

The following example creates a page with a floating frame, into which we load an ION Script page that displays an IDL-generated image.

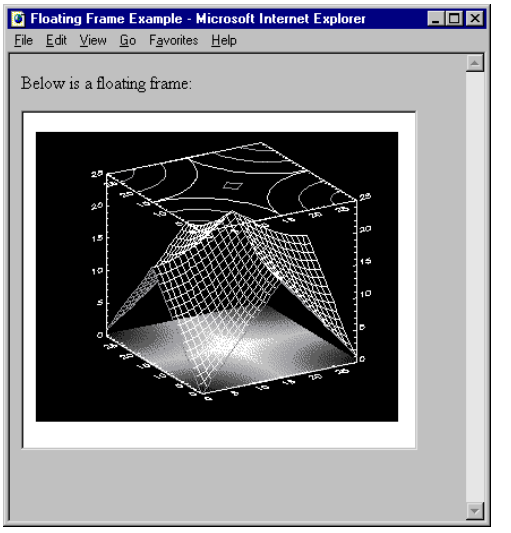
`msie.html`:

```
<HTML>
<HEAD>
   <TITLE>Floating Frame Example</TITLE>
</HEAD>
<BODY>
   Below is a floating frame:<BR><BR>
   <IFRAME NAME="Float1" SRC="ion-p?page=floatSRC.ion"
      WIDTH="350" HEIGHT="300">
   </IFRAME>
</BODY>
</HTML>
```

`floatSRC.ion:`

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IMAGE TYPE="DIRECT">
      <IDL>
         show3, dist(30)
      </IDL>
   </ION_IMAGE>
</ION_BODY>
</ION_SCRIPT>
```

When you load `msie.htm` in Internet Explorer, you'll see the following page:



Targeting Different Frames

Clicking a link, or a button or image with an associated event, causes another page to be loaded. By default, this new page replaces the current page in the full browser window. You may want the new page to be loaded in a different frame rather than in the full browser window. The frame or window in which the page is to be loaded is called the *target*. The ION_FORM, ION_IMAGE, and ION_LINK tags each have a TARGET attribute to specify the frame in which to load the URL specified by their corresponding EVENT attribute.

The TARGET attribute of ION_FORM, ION_IMAGE, and ION_LINK can be set to either the NAME attribute of an HTML <FRAME> or <IFRAME> tag, or to one of four predefined windows:

- _blank: opens the page in a new instance of the browser.
- _parent: opens the page in the next higher window in the window hierarchy (the immediate FRAMESET parent).
- _self: opens the page in the same window in which the event occurred. This is the default target if the TARGET attribute is not specified. This is useful for overriding a TARGET specified by the <BASE> tag.
- _top: opens the page in the main browser window.

Accommodating All Browsers

Keep in mind that different browsers have different capabilities with respect to frames. Floating frames, for example, are an Internet Explorer-specific feature. If you can't guarantee that users are viewing your page in Internet Explorer 3.0 or higher, it is best to either avoid floating frames, or to create a page that accommodates all browsers. The following page uses a floating frame for Internet Explorer, a <FRAMESET> for Netscape Navigator, and a <NOFRAMES> block for browsers that don't support frames:

```
<HTML>
<NOFRAMES>
   <FRAMESET COLS="50%,*">
      <FRAME SRC="ion-p?page=netscape1.html">
      <FRAME SRC="ion-p?page=netscape2.html">
   </FRAMESET>
   This content will not be visible to Internet Explorer <BR>
   or Netscape. Only non-frame browsers will display this.
</NOFRAMES>
<BODY>
   <IFRAME NAME="msie" SRC="ion-p?page=msie.html" WIDTH=150
      HEIGHT=150>
   </IFRAME>
   This text is visible in the BODY of the page in Internet
   Explorer.
</BODY>
</HTML>
```

Using JavaScript and VBScript

There are two ways to include JavaScript and VBScript in your ION Script applications:

- Use an HTML <SCRIPT>...</SCRIPT> block.
- Use an ION Script tag whose attribute takes a script as its value. The attributes that take scripts as values are any ON* attributes of the HTML <BODY> or <FORM> tags, such as ONLOAD and ONUNLOAD for <BODY>, and ONRESET and ONSUBMIT for <FORM>. These attributes can be set to either the name of a JavaScript/VBScript function or procedure or to a single JavaScript/VBScript statement. Note that different browsers support different sets of events.

  For example, to call the JavaScript function MyFunction() when the ION_FORM is submitted, you would set the ONSUBMIT attribute as follows:

  ```
  ONSUBMIT="MyFunction()"
  ```

  To pop up a message box saying “Form Submitted” when the form is submitted, you could set the ONSUBMIT attribute to a single JavaScript statement as follows:

  ```
  ONSUBMIT="alert('Form Submitted')"
  ```

Note

The “Advanced Examples” page, `index_examples.ion`, which can be accessed from the main examples page `index.ion`, contains several examples that demonstrate how to use JavaScript in an ION Script application, such as how to validate form data using JavaScript, and how to use JavaScript to capture mouse events in an ION_IMAGE configured as an image map.

Graphics in ION Script

ION Script images can be created using either of IDL's graphics systems: Direct Graphics or Object Graphics. The following sections describe how each graphics system is implemented in ION Script.

Direct Graphics

Images created using the IDL Direct Graphics commands are drawn to the IDL Z-Buffer. The Z-Buffer is an 8-bit device that stores intensity values for each pixel. These values are combined with the current color map to produce the final output image. If you change the device in any IDL code that is executed in an <ION_IMAGE> tag, you must make sure to change the device back to the Z-Buffer and TV the final image to it.

Object Graphics

ION Script creates a destination object on which to display Object Graphics trees. It is an RGB-mode IDLgrBuffer object. In any embedded ION Script code, it is referenced as ION_SCRIPT_BUFFER. Only graphics trees rendered to this instance of the object are displayed in the final Web page.

The following example illustrates how to use Object Graphics in ION Script. Note that it is not necessary, nor allowed, to create an IDLgrWindow object. To draw the object, you simply call the Draw method of the ION_SCRIPT_BUFFER object:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IMAGE TYPE="OBJECT">
      <IDL>
         file = FILEPATH('rose.jpg', SUBDIR=['examples', 'data'])
         READ_JPEG, file, image
         ;Create the objects:
         myview = OBJ_NEW('IDLgrView', view=[0,0,227,149])
         mymodel = OBJ_NEW('IDLgrModel')
         myimage = OBJ_NEW('IDLgrImage', image)
         ;Organize the object hierarchy:
         mymodel -> Add, myimage
         myview -> Add, mymodel
         ;Draw the view to the ION_SCRIPT_BUFFER:
         ION_SCRIPT_BUFFER-> Draw, myview
      </IDL>
   </ION_IMAGE>
</ION_BODY>
</ION_SCRIPT>
```

Bandwidth Issues

Because ION Script applications can be image-intensive, their performance depends strongly on network bandwidth. Bandwidth may not be an issue if you are serving your ION Script applications only to the users of your high-speed company intranet, but if your users are likely to be accessing your application over the Internet, through an analog telephone line and low-speed modem, then close consideration must be given to the file size of images generated by ION Script and to the overall design of your application. For example, if your application allows the user to zoom in on a region of interest, then you could provide the smallest, lowest-quality image necessary to give the user the required information at each stage in the drill-down process. If the final image delivered to the user is a 1200 x 1200 image with millions of colors, that does not necessarily mean that the images sent to the user to get to that point must also be so large. Another technique that can speed up your application is to use a page with several thumbnail images that the user can click on to see a more detailed, larger image.

Images can vary widely in file size, depending on the image format, the number of colors in the image, and the image dimensions. Image format and number of colors may or may not be factors that you can change, depending on your data and its intended use. If you can achieve acceptable results with a 256-color image instead of a 16.7-million color image, the price you pay in image detail and quality is rewarded with a smaller file size, and consequently, faster download time for the user. The image dimensions (i.e. the height and width of the image, in pixels), greatly effect file size. Use only as large an image as is necessary for your application.

Output Formats

Direct and Object Graphics images can be output to the Web browser in three different formats: 24-bit JPEG, 8-bit PNG, and 24-bit PNG.

The default format is 8-bit PNG for Direct Graphics images, and 24-bit JPEG for Object Graphics images, but these defaults can be changed via the configuration utility (see “Images” on page 18). The defaults can be overridden on a case-by-case basis by specifying a value for the IMG_TYPE attribute of the ION_IMAGE tag.

PNG uses a lossless compression scheme, whereas JPEG compression is lossy. JPEG compression can result in smaller image files, but the lossy compression may degrade image quality.

Whether you should use JPEG, 8-bit PNG, or 24-bit PNG depends on several factors. Some of these factors include:

- Whether your users have a browser that supports PNG. Assuming your user base is using a browser supported by ION Script (Netscape Navigator 4.7 or Internet Explorer 5.5), your users will have a browser that supports PNG.
- Whether you are using Direct or Object Graphics.
- The number of colors in the image.
- The required image quality.
- The resulting file size of the image.

The following are general guidelines for selecting the proper image format:

1. If your image is created using Direct Graphics, use 8-bit PNG. Since all Direct Graphics images created using ION Script are 8-bit, use 8-bit PNG instead of JPEG for Direct Graphics because JPEG compression of 8-bit images can result in degraded image quality. (See `image-types.ion` in the `examples` directory for an example of this.)
2. If your image is created using Object Graphics, the first factor to consider is the number of colors in your image. If your image contains 256 or fewer colors, use 8-bit PNG. If your image contains a large number of colors, use either 24-bit PNG or 24-bit JPEG. The choice comes down to how large the image file will be versus the required image quality. If application performance is of high importance and you can live with the visible artifacts that JPEG's lossy compression can introduce, JPEG might be a better choice because the file size of a JPEG image can be considerably smaller. The amount of degradation from JPEG compression varies depending on the image content, but should be acceptable for most Web applications. If image quality is of utmost importance, use 24-bit PNG.
3. A good way to determine which image format to use is to create the image in all three formats on the same page. You can then visually compare the images to see which formats give acceptable quality, and weigh the quality against the size of each image. To see the size of the image, right click on the image and select "Save image As" or "Save Picture As" to save the file. The example `image-types.ion` in the `examples` directory provides an example of how different image types affect the quality and size of the image. You can access this example by clicking the "Image Type Example" link on the "Basic" examples page, `index_howto.ion`.

Using Special and International Characters

ION supports 8-bit ASCII character sets as defined by ISO 8859, including ISO 8859-1 (also called Latin-1). This character set consists of 256 characters. The first 128 characters make up the US ASCII character set, consisting of characters used in US English. The second 128 characters support languages used in Western European countries.

Note

For a table of ISO 8859 character sets, visit http://www.bbsinc.com/iso8859.html.

In order to use 8-bit characters in an ION page using versions of ION prior to 1.2, it was necessary to insert either the character's *Numerical Entity Reference* or the *Entity Name*. The Numerical Entity Reference uses the format &#*nnn;*, where *nnn* is the decimal value of character. For example, the British pound symbol, £, can be inserted into an ION (or HTML) page using £. Some characters have been assigned an entity name to make it easier to remember the name for a character. For example, the British pound can be represented as &pound.

Note

Support for Entity Names is browser-dependent.

In ION, all ISO 8859-1 characters can be natively inserted into ION pages, directly from the keyboard. (Note that ION still supports Numerical Entity References and Entity Names.)

Example: Creating a Complete Application

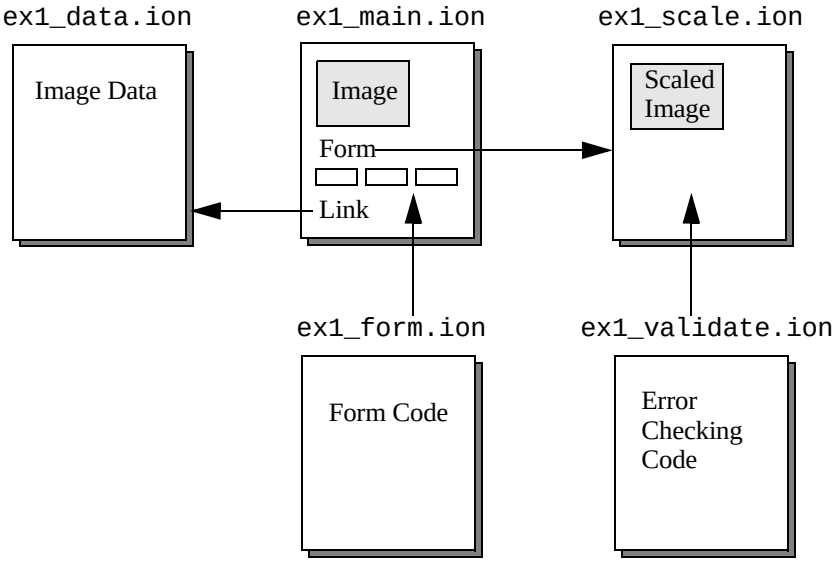
In this section, we will walk through the process of creating an interactive data-picking application. This application will do the following:

- Display a region of an IDL-generated image.
- Allow the user to input the desired color scale used to display the image.
- Check user input for invalid entries.
- Allow the user to select the center of the displayed region by clicking on the image.
- Provide a link that displays IDL-generated data about the image.

This application consists of five `.ion` files, located in the `examples` subdirectory of the ION Script installation. The following figure illustrates the relationship between the files we will be creating:

Note

The files for this example are included in the `examples` directory of your ION Script installation. You can load this example from the page `index_examples.ion` by clicking the "User's Guide Example Application" link.



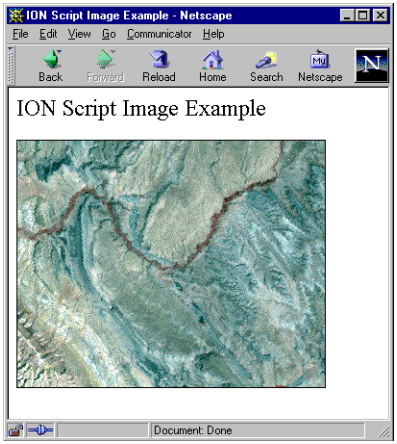
Step 1: Displaying an IDL-Generated Image

The first step is to create our main page, `ex1_main.ion`, on which we display an image. The original image is a 1200 x 1200 PNG image called `landsat.png`, which is located in the `ion_script/examples/data` subdirectory of your ION Script installation. Due to the bandwidth limitations of Web-based applications, we will display only a 320 x 256 region of the image. Another approach would be to scale the image to a smaller size.

The ION_IMAGE tag is used to insert an IDL-generated image into the page. The attributes of ION_IMAGE define basic image properties such as its width, height, and file format. The ION_IMAGE tag always contains an <IDL>...</IDL> block. The IDL tag can only be used inside the ION_IMAGE and ION_DATA_OUT tags. It is used inside an ION_IMAGE tag to define the IDL code used to generate an image, or inside an ION_DATA_OUT tag to insert textual output from IDL. In this example, we will enter the IDL code directly into the IDL block. We could also use the IDL tag to call an IDL `.pro` file instead. The following ION Script code displays a 320 x 256 region of the `landsat.png` image:

```
<ION_SCRIPT>
<ION_HEADER>
   <TITLE>ION Script Image Example</TITLE>
</ION_HEADER>

<ION_BODY>
   <FONT SIZE=+2>ION Script Image Example</FONT>
   <BR><BR>

   <ION_IMAGE WIDTH="320" HEIGHT="256" TYPE="DIRECT">
     <IDL>
       ; Read in the original image:
       image=read_png(filepath($
          subdir=['..', 'ion_script','examples', 'data'], $
          'landsat.png'), r, g, b)

       ; Create a 320x256 region of the image:
       region = image[440:759, 472:727]

       ; Load the display color table:
       tvlct, r, g, b

       ; Display the region:
       tv, region
     </IDL>
   </ION_IMAGE>
</ION_BODY>
</ION_SCRIPT>
```

This code creates the following Web page:


ION Script Image Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape
ION Script Image Example
Document: Done

Step 2: Declaring and Displaying Variables

In Step 1, we specified a subarray of our original image in order to display a smaller image on the Web page. To do this, we hard-coded the subscript values in the IDL command:

```
region = image[440:759, 472:727]
```

Suppose we want to use these subscript values in other parts of the application, or we want to allow the user to specify these subscript values. To do this, we can define variables to hold this information, and then reuse or change these variables as needed. Our code would be more flexible if we defined the region as follows:

```
region = image[xstart:xend, ystart:yend]
```

We can then define these IDL variables as follows:

xstart = Horizontal Center - 1/2 Region Width

xend = Horizontal Center + 1/2 Region Width - 1

ystart = Vertical Center - 1/2 Region Height

yend = Vertical Center + 1/2 Region Height - 1

Now we can define ION Script variables to hold the horizontal and vertical centers, and the region width and height. We want our region to be 320 x 256, and we'll place the center at the center of our 1200 x 1200 image, which makes the horizontal and vertical centers 600.

Declaring Variables

To define variables in an ION Script page, we use the VARIABLES tag, which contains one VARIABLE_DECL tag for each variable we declare. The VARIABLES tag resides inside the ION_HEADER. We'll add the following VARIABLES block to `ex1_main.ion`, in which we define the width and height of the original image, the region width and height, and the horizontal and vertical centers of the display region:

```
<VARIABLES>
   <VARIABLE_DECL NAME="IMAGE_W" VALUE="1200" TYPE="INT"/>
   <VARIABLE_DECL NAME="IMAGE_H" VALUE="1200" TYPE="INT"/>

   <VARIABLE_DECL NAME="REGION_W" VALUE="320" TYPE="INT"/>
   <VARIABLE_DECL NAME="REGION_H" VALUE="256" TYPE="INT"/>

   <VARIABLE_DECL NAME="CENTER_X" VALUE="600" TYPE="INT" />
   <VARIABLE_DECL NAME="CENTER_Y" VALUE="600" TYPE="INT" />
</VARIABLES>
```

Using Variables in ION Script Tags and IDL Code

We now need to insert the variables we have created into the code used to generate the image. To use a variable instead of an explicit value as an attribute of an ION Script tag or in IDL code, we precede the variable name with the $ symbol. This tells the ION Script parser to replace $*variable* with the value of *variable*. The following code uses our variables in the ION_IMAGE tag and in the IDL code:

```
<ION_IMAGE WIDTH="$REGION_W" HEIGHT="$REGION_H" TYPE="DIRECT">
   <IDL>
      ; Read in the original image:
      image = read_png(filepath($
         subdir=['..', 'ion_script', 'examples', 'data'], $
         'landsat.png'), r, g, b)

      ; Calculate the x subscripts:
      xstart = $CENTER_X - $REGION_W/2
      xend = $CENTER_X + $REGION_W/2 - 1

      ; Calculate the y subscripts:
      ystart = $CENTER_Y - $REGION_H/2
      yend = $CENTER_Y + $REGION_H/2 - 1

      ; Create the specified region:
      region = image[xstart:xend, ystart:yend]

      ; Load the display color table:
      tvlct, r, g, b
```

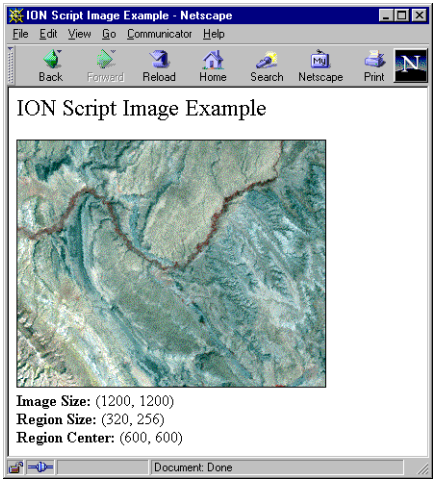
```
      ; Display the region:
      tv, region
   </IDL>
</ION_IMAGE>
```

Displaying Variable Values on the Page

At times, you will want to display the value of a variable on the Web page. For example, we could help the user of our application determine what portion of the image is being displayed by writing to the page the sizes of the original and displayed images, and the center of the display region. The ION_VARIABLE tag is used to insert the value of a variable into a page. We'll display three variables on our page by adding the following code to `ex1_main.ion`:

```
<BR>
<B>Image Size:</B> (<ION_VARIABLE NAME="$IMAGE_W"/>,
   <ION_VARIABLE NAME="$IMAGE_H"/>)<BR>
<B>Region Size:</B> (<ION_VARIABLE NAME="$REGION_W"/>,
   <ION_VARIABLE NAME="$REGION_H"/>)<BR>
<B>Region Center:</B> (<ION_VARIABLE NAME="$CENTER_X"/>,
   <ION_VARIABLE NAME="$CENTER_Y"/>)
```

Loading `ex1_main.ion` in our browser now results in the following page:


ION Script Image Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
ION Script Image Example
Image Size: (1200, 1200)
Region Size: (320, 256)
Region Center: (600, 600)
Document: Done

Note

The ION_VARIABLE tag can also be used to insert an ION Script variable into an HTML tag attribute. See "Variable Substitution in Attribute Values" on page 117.

Step 3: Adding Interactivity

For our application to be useful, we need to allow the user to control the way in which the data is presented. We'll add a form to our page to allow the user to specify the maximum and minimum pixel values to consider, and the maximum value of the scaled result. The form will include a button used to submit the user input and request a new page with the scaled image.

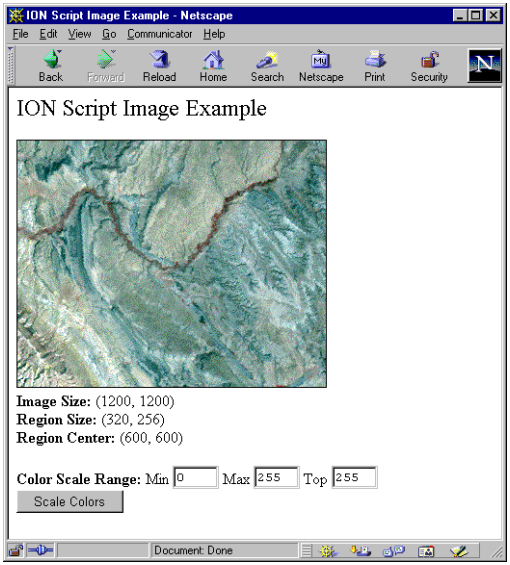
Creating an ION Script Form

Our interactive application will allow color value scaling of the image, for which we use the IDL BYTSCL function. This function takes the MIN, MAX, and TOP keywords, which we will allow the user to specify. Therefore, we need to get three values from the user.

User input in an ION Script application is achieved via an ION Script form, which is created with the ION_FORM tag. Our form will use three text input fields, created with the HTML INPUT tag. We also need a way to submit user input, which we'll accomplish with an ION_BUTTON. Add the following code to `ex1_main.ion` to create the form:

```
<ION_FORM>
   <B>Color Scale Range:</B>
   Min
   <INPUT TYPE="TEXT" NAME="SCALE_MIN" SIZE=4 VALUE="0">

   Max
   <INPUT TYPE="TEXT" NAME="SCALE_MAX" SIZE=4 VALUE="255">

   Top
   <INPUT TYPE="TEXT" NAME="SCALE_TOP" SIZE=4 VALUE="255">

   <BR>
   <ION_BUTTON TYPE="BUTTON" EVENT="SCALE_COLORS" LABEL="Scale
      Colors"/>
</ION_FORM>
```

Note that each text input field is given a name. A variable called $Form.*name* is automatically created for each text field, where *name* is the NAME attribute of the INPUT tag. Therefore, we can now access the values entered by the user by referring to the $Form.SCALE_MIN, $Form.SCALE_MAX, and $Form.SCALE_TOP variables. We assign each text field an initial value using the INPUT tag's VALUE attribute. This is the value that appears in the text field when the page is initially loaded. After adding a form, the page now looks like this:


ION Script Image Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
ION Script Image Example
Image Size: (1200, 1200)
Region Size: (320, 256)
Region Center: (600, 600)
Color Scale Range: Min 0 Max 255 Top 255
Scale Colors
Document: Done

Defining the Event

The ION_BUTTON tag in our form uses the EVENT attribute to specify the name of the event that occurs when the user clicks the button. In this case, we called our event "SCALE_COLORS". We must now define the SCALE_COLORS event, which we do by adding to the `ex1_main.ion` header block an EVENTS block that contains an EVENT_DECL tag:

```
<EVENTS>
   <EVENT_DECL NAME="SCALE_COLORS"
      ACTION="ion://ex1_scale.ion"/>
</EVENTS>
```

This EVENT_DECL specifies that the page `ex1_scale.ion` is processed when the SCALE_COLORS event occurs. In this case, the SCALE_COLORS event occurs when the user clicks the button. Note that the URL in the ACTION attribute of the EVENT_DECL tag begins with `ion://`. This causes the specified page to be processed by the ION Script parser. If you simply wanted to load an HTML page when the user clicks the button, you could use an `http://` or `file://` URL.

Creating Persistent Variables

We want the scaled image to use the same region and center as the original display region. Therefore, we need to pass the values of the REGION_W, REGION_H, CENTER_X, and CENTER_Y variables declared in `ex1_main.ion` to the `ex1_scale.ion` page. To do this, we must make the variables persistent by specifying the PERSIST attribute in our VARIABLE_DECL tags as follows:

```
<VARIABLE_DECL NAME="REGION_W" VALUE="320" TYPE="INT"
   PERSIST="TRUE"/>
<VARIABLE_DECL NAME="REGION_H" VALUE="256" TYPE="INT"
   PERSIST="TRUE"/>

<VARIABLE_DECL NAME="CENTER_X" VALUE="600" TYPE="INT"
   PERSIST="TRUE"/>
<VARIABLE_DECL NAME="CENTER_Y" VALUE="600" TYPE="INT"
   PERSIST="TRUE"/>
```

Note

We do not need to declare persistent variables for the values entered by the user. Values entered in a form become $Form variables, which are automatically passed to the action page when an event occurs.

Creating the Action Page

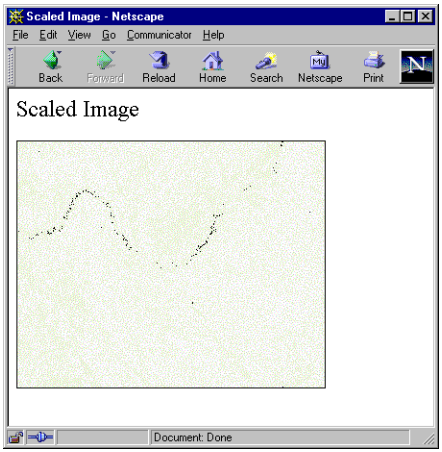
We've provided a means of user input, established the variables as persistent so that their values can be passed to the action page, and specified that the page `ex1_scale.ion` be processed when the user clicks the button. All that's left is to create the `ex1_scale.ion` page that generates the scaled image.

We can make use of the $Form variables in the IDL code that generates the scaled image. The `ex1_scale.ion` page is coded as follows:

```
<ION_SCRIPT>

<ION_HEADER>
   <TITLE>Scaled Image</TITLE>
</ION_HEADER>

<ION_BODY>
   <FONT SIZE="+2">Scaled Image</FONT><BR><BR>
   <ION_IMAGE  WIDTH="$REGION_W" HEIGHT="$REGION_H" TYPE="DIRECT">
      <IDL>
         ; Read in the original image:
         image = read_png(filepath($
            subdir=['..', 'ion_script', 'examples', 'data'],$
            'landsat.png'), r, g, b)
```

```
            ; Calculate the x subscripts:
            xstart = $CENTER_X - $REGION_W/2
            xend = $CENTER_X + $REGION_W/2 - 1

            ; Calculate the y subscripts:
            ystart = $CENTER_Y - $REGION_H/2
            yend = $CENTER_Y + $REGION_H/2 - 1

            ; Create the region to be displayed:
            region = image[xstart:xend, ystart:yend]

            ; Scale the region:
            scale_region = bytscl(region, MIN=$Form.SCALE_MIN,$
               MAX=$Form.SCALE_MAX, TOP=$Form.SCALE_TOP)

            ; Load the display color table:
            tvlct, r, g, b

            ; Display the region:
            tv, scale_region
        </IDL>
    </ION_IMAGE>
</ION_BODY>

</ION_SCRIPT>
```

After we enter some values in the form and click the button, we get the following page:


Scaled Image - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Scaled Image
Document: Done

Step 4: Validating Form Data

To properly scale the image in our example, the values you enter in any of the form fields must be between 0 and 255, and the value you enter for SCALE_MIN must be smaller than the value for SCALE_MAX. We can make our application more user-friendly by checking the data entered by the user, and reporting any invalid entries. To do this, we need to be able to conditionally execute portions of our code. For example, to check for invalid SCALE_MIN values, we'd use an algorithm such as:

> If SCALE_MIN is less than 0
>
> Write "Scale min must be greater than 0" to the page
>
> If SCALE_MIN is greater than 255
>
> Write "Scale min must be less than 255" to the page

We can access the values submitted in the form with the variables $Form.SCALE_MIN, $Form.SCALE_MAX, and $Form.SCALE_TOP. But $Form variables are always treated as strings, therefore we cannot use $Form variables in numeric expressions. In order to check the numeric values submitted in the form, we must first declare numeric variables, and set them equal to the $Form variables. This converts the string to a numeric value. We'll therefore add the following VARIABLES block to the `ex1_scale.ion` page:

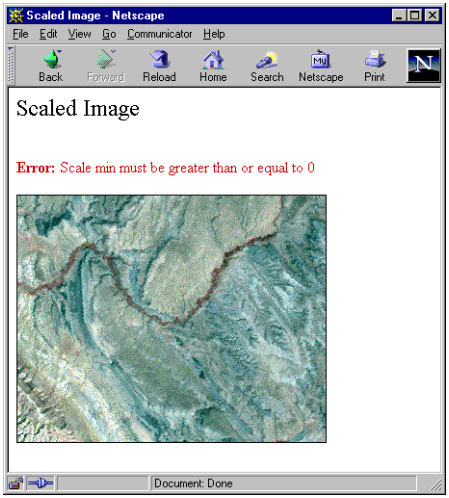
```
<VARIABLES>
   <VARIABLE_DECL NAME="SCALE_MIN" VALUE="$Form.SCALE_MIN"
      TYPE="INT" />
   <VARIABLE_DECL NAME="SCALE_MAX" VALUE="$Form.SCALE_MAX"
      TYPE="INT" />
   <VARIABLE_DECL NAME="SCALE_TOP" VALUE="$Form.SCALE_TOP"
      TYPE="INT" />
</VARIABLES>
```

Now we're ready to check the values submitted in the form. We do this by creating an ION_IF block. For example, we can check the value of SCALE_MIN by adding the following code to `ex1_scale.ion`:

```
<ION_IF EXPR="$SCALE_MIN LT 0">
   <FONT COLOR="#CC0000"><BR>
   <B>Error:</B> Scale min must be greater than or equal to 0
   <BR><BR>
   </FONT>
<ION_ELSEIF EXPR="$SCALE_MIN GT 255"/>
   <FONT COLOR="#CC0000"><BR><BR>
   <B>Error:</B> Scale min must be less than 256 <BR>
   </FONT>
</ION_IF>
```

For more on ION_IF, see “ION_IF, ION_ELSEIF, ION_ELSE” on page 158.

Now if the user submits a negative value for SCALE_MIN, an error will be written to the page, as shown in the following figure:


Scaled Image - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Scaled Image
Error: Scale min must be greater than or equal to 0
Document: Done

The scaled image, however, will still be generated with invalid data. Therefore, if any values are invalid, we need to change them to valid values. To change the value of a variable on the fly, we use the ION_EVALUATE statement as follows:

```
<ION_IF EXPR="$SCALE_MIN LT 0">
   <ION_EVALUATE EXPR="$SCALE_MIN = 0"/>
   <FONT COLOR="#CC0000"><BR>
   <B>Error:</B> Scale min must be greater than or equal to 0
   <BR>
   </FONT>
<ION_ELSEIF EXPR="$SCALE_MIN GT 255"/>
   <ION_EVALUATE EXPR="$SCALE_MIN = 255"/>
   <FONT COLOR="#CC0000"><BR>
   <B>Error:</B> Scale min must be less than 256 <BR>
   </FONT>
</ION_IF>
```

Step 5: Creating Reusable Pages

So far, this application uses only one image. In a real-world application, we would most likely want to be able to perform the same manipulations on multiple images. We could save ourselves from rewriting much of the code by breaking off reusable portions of code into separate files, and then *including* these files. To include a file, we use the ION_INCLUDE tag.

First, we'll cut the ION_FORM block from `ex1_main.ion`, and include a separate ION Script page, which we'll call `ex1_form.ion`. To do this, we replace the ION_FORM tag in `ex1_main.ion` with ION_INCLUDE as follows:

```
<ION_INCLUDE SRC="ion://ex1_form.ion" />
```

The include file looks like this:

```
<ION_FORM>
   <B>Color Scale Range:</B>
   Min
   <INPUT TYPE="TEXT" NAME="SCALE_MIN" SIZE=4
      VALUE="0">
   Max
   <INPUT TYPE="TEXT" NAME="SCALE_MAX" SIZE=4
      VALUE="255">
   Top
   <INPUT TYPE="TEXT" NAME="SCALE_TOP" SIZE=4
      VALUE="255">
   <BR>
   <ION_BUTTON TYPE="BUTTON" EVENT="SCALE_COLORS"
      LABEL="Scale Colors"/>
</ION_FORM>
```

Note that we do not need to use the ION_SCRIPT or ION_BODY tags in this include file. The file is simply inserted into the body of the file containing the ION_INCLUDE tag.

Our application also places all of the error-checking code into its own ION Script page called `ex1_validate.ion`, and uses the following tag in `ex1_scale.ion`:

```
<ION_INCLUDE SRC="ion://ex1_validate.ion" />
```

Step 6: Creating Interactive Images for Data Picking

Two useful features of ION Script are its ability to assign events to images, and to capture the location on an image where the mouse was clicked. These features allow us to create powerful data-picking applications.

Since our example application displays only a portion of the original image, we could allow the user to select the portion of the image to view. We'll add an event to the image on the `ex1_main.ion` page that causes the image to be re-centered at the location of the mouse click.

First, we need to define an event for the image. In the EVENTS block of `ex1_main.ion`, we define the event as follows:

```
<EVENT_DECL NAME="CENTER_IMAGE"
   ACTION="ion://ex1_main.ion"/>
```

Notice that the action of this event is to reload the same page. Now we need to add an EVENT attribute to the image itself. The ION_IMAGE tag in `ex1_main.ion` becomes:

```
<ION_IMAGE EVENT="CENTER_IMAGE" WIDTH="$REGION_W"
   HEIGHT="$REGION_H" TYPE="DIRECT">
```

Using the $Mouse System Variable

Next, we need to add code to find the new center, and make sure that we haven't reached one of the four edges of the image. Our tool for finding the location of the mouse click is the $Mouse system variable. $Mouse.x stores the x location of the mouse click, and $Mouse.y stores the y location. Because our image is always 320 x 256, the display center cannot be less than 320/2 pixels away from the left or right edges of the original image, or 256/2 pixels away from the top or bottom edges. Therefore, we must also add error-checking code to adjust the display center if the user clicks outside these boundaries. We'll add the following code to `ex1_main.ion` to recalculate the CENTER_X and CENTER_Y variables:

```
<!-- If the mouse was pressed, recompute the center -->
<ION_IF EXPR="$Mouse.x NE 0.0">
   <ION_EVALUATE
      EXPR="$CENTER_X = $CENTER_X + ($Mouse.x - $REGION_W/2)"/>
   <ION_EVALUATE
      EXPR="$CENTER_Y = $CENTER_Y - ($Mouse.y - $REGION_H/2)"/>
</ION_IF>

<!-- Make sure we haven't reached the edge of the image -->
<ION_IF EXPR="($CENTER_X + $REGION_W/2) GT ($IMAGE_W - 1)">
   <ION_EVALUATE EXPR="$CENTER_X = ($IMAGE_W - 1) - $REGION_W/2"/>
   <FONT COLOR="#0000CC">
```

```
   <BR>Right edge of image reached, view center has been adjusted
   </FONT>
<ION_ELSEIF EXPR="($CENTER_X - $REGION_W/2) LT 0"/>
   <ION_EVALUATE EXPR="$CENTER_X = $REGION_W/2"/>
   <FONT COLOR="#0000CC">
   <BR>Left edge of image reached, view center has been adjusted
   </FONT>
</ION_IF>

<ION_IF EXPR="($CENTER_Y + $REGION_H/2) GT ($IMAGE_H - 1)">
   <ION_EVALUATE EXPR="$CENTER_Y = ($IMAGE_H - 1) - $REGION_H/2"/>
   <FONT COLOR="#0000CC">
   <BR>Top edge of image reached, view center has been adjusted
   </FONT>
   <ION_ELSEIF EXPR="($CENTER_Y - $REGION_H/2) LT 0"/>
   <ION_EVALUATE EXPR="$CENTER_Y = $REGION_H/2"/>
   <FONT COLOR="#0000CC">
   <BR>Bottom edge of image reached, view center has been adjusted
   </FONT>
</ION_IF>
```

Step 7: Displaying IDL-Generated Data

Our final task for this application is to provide the user with some data about the image. We'll add a link to the `ex1_main.ion` page that executes a page called `ex1_data.ion`, which displays the minimum, maximum, and mean values in the image.

To create the link, we use the ION_LINK tag as follows:

```
<ION_LINK EVENT="SHOW_DATA">View Image Data</ION_LINK>
```

Like an ION_BUTTON, an ION_LINK has an EVENT attribute. When the user clicks on the link, we want to load `ex1_data.ion`. We'll therefore define the event as follows:

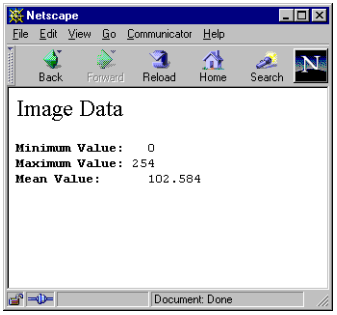
```
<EVENT_DECL NAME="SHOW_DATA" ACTION="ion://ex1_data.ion"/>
```

Next, we need to create the `ex1_data.ion` page. Whereas the ION_IMAGE tag displays an IDL-generated image, the ION_DATA_OUT tag is used to insert textual IDL output into the page. The following code is used to create the `ex1_data.ion` page:

```
<ION_SCRIPT>
<ION_HEADER>
   <TITLE>Image Data</TITLE>
</ION_HEADER>
<ION_BODY>
   <FONT SIZE="+2">Image Data</FONT>
```

```
<BR>
<ION_DATA_OUT>
   <IDL>
      ; Read in the original image:
      image = read_png(filepath($
         subdir=['..', 'ion_script', 'examples', 'data'],$
         'landsat.png'), r, g, b)

      ; Calculate the x subscripts:
      xstart = $CENTER_X - $REGION_W/2
      xend = $CENTER_X + $REGION_W/2 - 1

      ; Calculate the y subscripts:
      ystart = $CENTER_Y - $REGION_H/2
      yend = $CENTER_Y + $REGION_H/2 - 1

      ; Create the region to be displayed:
      region = image[xstart:xend, ystart:yend]

      ; Print min, max, and mean:
      print, '<B>Minimum Value:</B>', min(region)
      print, '<B>Maximum Value:</B>', max(region)
      print, '<B>Mean Value:</B>', mean(region)
   </IDL>
</ION_DATA_OUT>
</ION_BODY>
</ION_SCRIPT>
```

Note that the IDL print statements include HTML. The IDL output inserted by ION_DATA_OUT is inserted directly into the HTML stream, therefore, we can include HTML in our IDL print statements to control the way the IDL output is rendered by the browser. The following figure shows the page generated by `ex1_data.ion`:



Note

For another example of how ION_DATA_OUT can be used to generate HTML, see "ION_DATA_OUT" on page 141. Also see the examples `dropframe.ion` and `pickfile.ion`, which can be accessed by the "Dynamic Droplist Example" and "Dynamic Table Example" links on the Advanced examples page, `index_examples.ion`.

A Note On Using the Back Button

When you design an application, always test the behavior of the application when navigating with the browser's Back button. Using the Back button to navigate through an application may cause undesired results. Try to design your application interface to discourage use of the Back button. For example, on the `ex1_scale.ion` and `ex1_data.ion` pages, you must use the Back button to return to the previous page. We could add an ION_BUTTON to each page and assign an event that returns to the previous page. This ensures that any changed variable values are passed back to the previous page. Note, however, that you cannot depend on the user clicking your ION_BUTTON instead of the Back button. Therefore, make sure that using the Back button does not cause problems with your application, regardless of whether you provide alternative means of navigation.

Application Variations

To keep this example as simple as possible, our button event caused the action page to replace the main page. In this section, we'll explore a couple alternative ways of writing this application.

Using Frames

In some applications, you might find it useful to display an original and a manipulated image next to each other on the same Web page. One way to do this would be to use frames. We've included a framed version of this example in the `examples` directory. The file is called `ex1_frame_main.html`. Load this page using a URL such as

```
http://myserver/cgi-bin/ion-p?page=ex1_frame_main.html
```

For more information on creating frames, see "Using Frames with ION Script" on page 82.

Reloading the Same Page

Another option for this application would be to use the same page for the main page and the action page. In this case, when the user clicks the button, the action is to reload the same page. We've included this version in the `examples` directory. The file is called `ex2_main.ion`.

There are a few changes we need to make to the application when the event is to reload a page with a form:

1. First, we need to change the event declaration to load the same page:

```
<EVENT_DECL NAME="SCALE_COLORS" ACTION="ion://ex2_main.ion"/>
```

2. The first time the page is loaded, the image is scaled using the values defined for SCALE_MIN, SCALE_MAX, and SCALE_TOP in the VARIABLE_DECL tags. When the user enters new values in the form and clicks the button, we want to use the values entered by the user rather than the values defined in the VARIABLE_DECL tags. Therefore, we need one set of variables that set the scale values when the page is initially loaded (or loaded from a form other than the one containing the entries), and another set of variables to set the scale values from the form:

```
<VARIABLE_DECL NAME="SCALE_MIN" VALUE="0" TYPE="INT"
   PERSIST="TRUE"/>
<VARIABLE_DECL NAME="SCALE_MAX" VALUE="255" TYPE="INT"
   PERSIST="TRUE"/>
<VARIABLE_DECL NAME="SCALE_TOP" VALUE="255" TYPE="INT"
   PERSIST="TRUE"/>

<VARIABLE_DECL NAME="USER_SCALE_MIN"
   VALUE="$Form.USER_SCALE_MIN" TYPE="INT" />
<VARIABLE_DECL NAME="USER_SCALE_MAX"
   VALUE="$Form.USER_SCALE_MAX" TYPE="INT" />
<VARIABLE_DECL NAME="USER_SCALE_TOP"
   VALUE="$Form.USER_SCALE_TOP" TYPE="INT" />
```

3. When we reload the page `ex2_main.ion`, we'd like the values that appear in the form fields to remain the same as the values entered by the user the last time the "Scale Colors" button was clicked. Therefore, we don't want to use the hard-coded values that `ex1_form.ion` uses. Instead, we'll use ION_VARIABLE to insert the user value into the form fields. Also, the original form uses SCALE_MIN, SCALE_MAX, and SCALE_TOP as the names of the form fields. We want to use the user values instead, so we'll change these names to USER_SCALE_MIN, USER_SCALE_MAX. and USER_SCALE_TOP. We'll save these changes to the form in a new file called `ex2_form.ion`, which now looks like this:

```
<ION_FORM>
      <B>Color Scale Range:</B>
Min
   <INPUT TYPE="TEXT" NAME="USER_SCALE_MIN" SIZE=4
      VALUE="<ION_VARIABLE NAME="$SCALE_MIN"/>">
Max
   <INPUT TYPE="TEXT" NAME="USER_SCALE_MAX" SIZE=4
      VALUE="<ION_VARIABLE NAME="$SCALE_MAX"/>">
Top
   <INPUT TYPE="TEXT" NAME="USER_SCALE_TOP" SIZE=4
      VALUE="<ION_VARIABLE NAME="$SCALE_TOP"/>">
   <BR>
   <ION_BUTTON TYPE="BUTTON" EVENT="SCALE_COLORS" LABEL="Scale
      Colors"/>
</ION_FORM>
```

4. In order to use the new form page, `ex2_form.ion`, we need to change the ION_INCLUDE tag that includes the form:

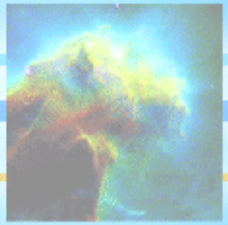
```
<ION_INCLUDE SRC="ion://ex2_form.ion"/>
```

5. Next, we need to retrieve the user values from the form. We do this by adding the following code to `ex2_main.ion`:

```
<ION_IF EXPR="$Form.USER_SCALE_MIN NE 'undef'">
    <ION_EVALUATE EXPR="$SCALE_MIN = $USER_SCALE_MIN"/>
    <ION_EVALUATE EXPR="$SCALE_MAX = $USER_SCALE_MAX"/>
    <ION_EVALUATE EXPR="$SCALE_TOP = $USER_SCALE_TOP"/>
</ION_IF>
```

6. Lastly, we need to add an ION_INCLUDE tag in `ex2_main.ion` to include the validation page:

```
<!-- Validate the user input -->
<ION_INCLUDE SRC="ion://ex1_validate.ion"/>
```



Chapter 5: ION Script Tag Reference

The following topics are covered in this chapter:

- Syntax Conventions
- HTML Mappings
- The HTML Comment Tag
- Alphabetical Listing of ION Script Tags

Syntax Conventions

ION Script tags take one of the following two forms:

- **Tag Pair:** <TAG> *content* </TAG>
- **Single Tag:** <TAG/>

Note

It is important to note that single tags are closed with />. Don't forget the /, or you will get a syntax error.

The following table lists which ION Script tags are tag pairs, and which are single tags:

Tag Pairs	Single Tags
<EVENTS> ... </EVENTS>	<EVENT_DECL />
<IDL> ... </IDL>	<ION_BUTTON />
<ION_BODY> ... </ION_BODY>	<ION_CHECKBOX />
<ION_DATA_OUT> ... </ION_DATA_OUT>	<ION_EVALUATE />
<ION_FORM> ... </ION_FORM>	<ION_ELSE />
<ION_HEADER> ... </ION_HEADER>	<ION_ELSEIF />
<ION_IF> ... </ION_IF>	<ION_INCLUDE />
<ION_IMAGE> ... </ION_IMAGE>	<ION_PARAM />
<ION_LINK> ... </ION_LINK>	<ION_RADIO />
<ION_OBJECT> ... </ION_OBJECT>	<ION_VARIABLE />
<ION_SCRIPT> ... </ION_SCRIPT>	<VARIABLE_DECL />
<VARIABLES> ... </VARIABLES>	

Elements of Syntax

Each tag listed in this reference includes a “Syntax” section that defines how the tag is used. The following elements are used in syntax listings:

Element	Description
[ ] (Square brackets)	Indicates that the contents are optional.
Italics	Indicates arguments, expressions, or statements for which you must provide values.
{ } (Braces)	Indicates that you must choose one of the values they contain.
| (Vertical lines)	Separates values from which you must choose.

Consider the following syntax example:

<TAG ATTRIBUTE1="*value"* [ATTRIBUTE2="*value"*]
[ATTRIBUTE3={"*value1*" | "*value2*"}] >

content

</TAG>

In this example, ATTRIBUTE1 is required, while all other attributes are optional. ATTRIBUTE3 can be set to either *value1* or *value2*.

Square Brackets ([ ])

- Content between square brackets is optional.
- Do not include square brackets in your statement.

Note

Square Brackets are used in IDL to define an array. When including IDL code as the value of an ION Script tag attribute or the content of an ION Script tag, square brackets that are part of the IDL code must be included. For example, the <IDL> tag might be used as follows:

```
<IDL> MyArray = [1,2,3,4,5] </IDL>
```

In this case, the square brackets are included in the statement because they are part of the IDL syntax.

Braces ({ })

- Braces are used to enclose a list of the possible values for an attribute. You must choose one of the listed values, which are separated by a vertical line (|). Do not include the braces in your statement. For example, the following syntax indicates that you must choose either TRUE or FALSE for the DISPLAY attribute (if you choose to use DISPLAY):

  <ION_EVALUATE EXPR="*expression*" [DISPLAY={"TRUE" | "FALSE"}] />

  An example of a valid statement is:

  ```
  <ION_EVALUATE EXPR="$X * 5" DISPLAY="TRUE" />
  ```

Note

Braces are used in IDL to define a structure. When including IDL code as the value of an ION Script tag attribute or the content of an ION Script tag, braces that are part of the IDL code must be included. For example, the <IDL> tag might be used as follows:

```
<IDL> MyStructure = {a:1, b:2, c:3} </IDL>
```

In this case, the braces are included because they are part of the IDL syntax.

Italics

- Italicized words are place-holders for values that you must provide for attributes, as well as for HTML or IDL content.

Case Sensitivity

ION Script is case-sensitive. All ION Script tags and attributes must be entered exactly as shown in this reference. User-defined variables must use consistent case as well. If you declare a variable as $A, you cannot refer to that variable using $a. System variables are not case-sensitive, except for $Browser system variables.

Using Quotation Marks

ION Script requires that values assigned to tag attributes be enclosed in double quotation marks, using the general form ATTRIBUTE="*value*".

Note

Carriage returns are not allowed between the double quotation marks that delimit the value of an ION Script attribute.

When the attribute is an expression (the EXPR attribute of ION_DATA_OUT or ION_IF), you will often need to mix numeric and string values. Strings can be

enclosed in single quotation marks, which can be nested inside double quotation marks, as in the following example:

```
<ION_IF EXPR="$Name EQ 'Smith' ">
```

Single quotation marks must also be nested inside double quotation marks whenever you assign a string literal as the value of an ION Script variable. For example, the following code shows the correct way to assign a string literal to the VALUE attribute of the VARIABLE_DECL tag:

```
<VARIABLE_DECL NAME="animal" TYPE="STR" VALUE="'dog'"/>
```

Note

In an <IDL> block, you can use either single or double quotation marks to delimit a string. Note, however, that in IDL, a double quotation mark followed by a number has a special meaning. See "Single vs. Double Quotation Marks" on page 126.

Variable Substitution in Attribute Values

Many ION Script tag attributes can be set to values that contain explicit values and/or variables. For example, the EVENT attribute could be set in the following ways:

- To an explicit value:

  ```
  EVENT="http://myserver/page1.html"
  ```

- To a variable:

  ```
  EVENT="$MyURL"
  ```

- To a combination of explicit values and variables:

  ```
  EVENT="http://myserver/$mypage"
  ```

If an attribute can contain a variable, this fact is indicated in the attribute description.

HTML tag attributes can also contain variables, but you must insert the variable into the HTML attribute by using the ION_VARIABLE tag. For example, the HTML <FONT> tag could be specified as follows:

```
<ION_HEADER>
<VARIABLES>
   <VARIABLE_DECL NAME="FontColor" TYPE="STR" VALUE="'Red'"/>
</VARIABLES>
</ION_HEADER>
<ION_BODY>
   <FONT COLOR="<ION_VARIABLE NAME="$FontColor"/>">
   This should be Red
   </FONT>
</ION_BODY>
```

HTML Mappings

When an ION script is parsed, ION Script tags and attributes are converted to HTML. HTML tags, as well as HTML attributes used in ION Script tags, are passed through to the browser unchanged. This allows you to use any valid HTML in your ION scripts, and ensures that you can always make use of the latest HTML features as Web browsers and the HTML language evolve.

By knowing what HTML is produced by each ION Script tag, you can include in your ION Script tag any attributes associated with the corresponding HTML tag. For example, the ION_BUTTON tag, when used with the TYPE="IMAGE" attribute, is converted to the following HTML:

```
<INPUT TYPE="IMAGE" BORDER="1">
```

Knowing that the <ION_BUTTON> tag becomes an HTML <INPUT> tag allows you to include in your <ION_BUTTON> tag any valid attributes of the <INPUT> tag. For example, you could include the ALIGN attribute to the <INPUT> tag as follows:

```
<ION_BUTTON
   EVENT="myevent"
   SRC="http://www.ResearchSystems.com/images/ion_sm.png"
   ALIGN="middle"
   TYPE="IMAGE"/>
```

For each ION Script tag documented in this reference, you will find a section called “HTML Mapping”. The “HTML Mapping” section for each ION Script tag specifies the HTML that is produced by that tag, if any. You can then consult an HTML reference to get a list of all the attributes that the HTML tag supports.

Note that certain ION Script tag attributes are converted to HTML attributes. You cannot directly specify an HTML attribute if that attribute is created from an ION Script attribute. For example, the ION Script code

```
<ION_BUTTON EVENT="event" TYPE="BUTTON"/>
```

is converted to the HTML

```
<INPUT TYPE="SUBMIT" NAME="event"/>
```

Therefore, if you specify the NAME attribute in the ION_BUTTON tag, it will be ignored.

The HTML Comment Tag

The HTML comment tag is used to add text to an HTML document without including that text in the Web page. When used in an ION Script page, HTML comment tags and the content they enclose are passed unchanged directly to the browser.

The HTML comment tag consists of the opening tag <!-- and the closing tag -->. Text between <!-- and --> will not appear in the Web page, nor will it be evaluated by the ION Script parser:

```
<!-- This text will not be displayed on the Web page -->
```

This comment will, however, be passed in the HTML file sent to the browser, and can be seen by viewing the page source (**View → Source**).

Comments can be used on a single line as shown above, or can span multiple lines as follows:

```
<!-- This text will not be
displayed on the Web page -->
```

Comments are used in the following ways:

- Documenting your code. Comments are useful in documenting your Web page or ION Script application, and can help clarify your ION Script code for you and other developers.
- Debugging ION Script applications. Comments allow you to “turn off” selected ION Script tags by commenting them out, in order to test individual code segments. For example, to comment out an ION_EVALUATE tag, you would use the comment tag as follows:

  ```
  <!-- <ION_EVALUATE EXPR="$A = 10"/> -->
  ```

- Prevent content inside HTML tags from appearing as text in older browsers that do not recognize the tag. For example, the <STYLE> tag is not recognized by some older browsers, in which case the content inside the <STYLE> tag appears as text. The HTML comment tag is often used inside the <STYLE> tag to prevent style declarations from appearing as text in older browsers:

  ```
  <STYLE type="text/css"><!--
     H1 {color: red;}
     BODY {background: yellow;}
  --></STYLE>
  ```

  Similarly, because older browsers do not support scripting languages such as JavaScript and VBScript, HTML comment tags are often used inside <SCRIPT> blocks to prevent scripts from appearing as text in older browsers:

```
<SCRIPT TYPE="text/javascript">
<!--
   document.write("Hello World");
//-->
</SCRIPT>
```

Note

IDL code inside an <IDL>...</IDL> block can also contain comments, but IDL comment syntax is used in this case, not the HTML comment tag. For information on using comments in IDL code, see “Commenting IDL Code” on page 127.

Alphabetical Listing of ION Script Tags

This section contains an alphabetical listing of all ION Script tags, including their syntax, and a description of each attribute. The following ION Script tags are covered in this section:

Tag	Page
EVENT_DECL	122
EVENTS	123
IDL	124
INPUT	129
ION_BODY	133
ION_BUTTON	135
ION_CHECKBOX	138
ION_DATA_OUT	141
ION_EVALUATE	149
ION_FORM	152
ION_HEADER	156
ION_IF, ION_ELSEIF, ION_ELSE	158
ION_IMAGE	159
ION_INCLUDE	165
ION_LINK	170
ION_OBJECT	173
ION_PARAM	178
ION_RADIO	179
ION_SCRIPT	182
ION_VARIABLE	183
VARIABLE_DECL	186
VARIABLES	189

EVENT_DECL

The EVENT_DECL tag declares an event. When the user clicks a button, image, or link with an assigned event, the page defined by the event is processed and displayed. The event name is used as the value of the EVENT attribute of the ION_BUTTON, ION_IMAGE, and ION_LINK tags.

Note

The EVENT_DECL tag must reside between the <EVENTS> and </EVENTS> tags.

Syntax

<EVENT_DECL
NAME="*name*{A-Z, a-z, 0-9, and _ only}"
ACTION="*url*" />

Attributes

ACTION

Defines the URL of the document to be displayed when the event occurs. See "Specifying URLs" on page 70 for more on specifying valid URLs

NAME

Defines the name of the event. This name is used as the value for the EVENT attribute of the ION_BUTTON, ION_IMAGE, and ION_LINK tags. The value specified for the NAME attribute can only contain the characters A-Z, a-z, 0-9 or _, and cannot be the name 'page'.

HTML Mapping

The EVENT_DECL tag does not produce HTML and therefore does not support any additional attributes.

EVENTS

The EVENTS tag pair is used to delimit a block of event declarations in the header of an ION Script document. The <EVENTS>...</EVENTS> block can contain one or more EVENT_DECL tags.

Note

The EVENTS tag pair must reside between the <ION_HEADER> and </ION_HEADER> tags.

Syntax

<EVENTS>

EVENT_DECL tags

</EVENTS>

Attributes

None

HTML Mapping

The EVENT tag does not produce HTML and therefore does not support any additional attributes.

IDL

The IDL tag pair delimits a block of IDL code. This tag pair must reside inside an ION_DATA_OUT or ION_IMAGE tag pair. Any valid IDL code is allowed between <IDL> and </IDL>, with the exception of the items listed below under “IDL Tag Limitations”.

Syntax

<IDL>

[ION_EVALUATE or ION_VARIABLE tags]

IDL code (can contain $*variable*)

</IDL>

Note

The IDL tag pair must reside inside an ION_IMAGE or ION_DATA_OUT tag pair.

Attributes

None

IDL Tag Limitations

The following IDL code cannot be used inside an <IDL>...</IDL> block:

- You cannot use the OBJ_NEW function to create a window. There is no need to create your own IDLgrWindow object. Instead, you draw an object by calling the Draw method on the ION_SCRIPT_BUFFER object. See “Object Graphics” on page 91 for an example of how to use Object Graphics in ION Script. To run an example, run the example file `object-graphics.ion` in the `examples` directory.
- You should not use the WINDOW procedure to create a graphics or text window. ION Script sets up the drawing area itself.
- Widgets and compound widgets are not available in ION Script.
- You cannot use the <IDL> block to write programs as you would in a `.pro` file. ION Script sends the commands in the <IDL> block to IDL one line at a time. Therefore, each line inside the <IDL> block must be a statement that is valid at the IDL command line. For example, you *cannot* use an <IDL> block as follows:

```
<IDL>
   FOR X=1,20 DO BEGIN
     PRINT, X
   ENDFOR
</IDL>
```

To perform the above statements, you would need to write a `.pro` file, and call the program from the <IDL> block.

Using ION_EVALUTE and ION_VARIABLE Tags in an IDL Block

When an IDL block is evaluated by the ION Script parser, the following sequence of events occurs:

1. All ION_EVALUATE and ION_VARIABLE blocks are evaluated in the order in which they appear in the IDL block, and the result of each block replaces the ION Script code in the IDL block.
2. ION Script variables specified by $*variable* are evaluated and the result replaces the ION Script variable in the IDL block.

The following example illustrates a case in which you might need to have the ION Script parser evaluate an ION_EVALUATE block before passing the data to IDL:

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="A" VALUE="123" TYPE="INT"/>
   </VARIABLES>
</ION_HEADER>
<ION_BODY>
<ION_DATA_OUT METHOD="GET">
   <IDL>
      help, $A
      help, <ION_VARIABLE NAME="$A" FORMAT="%dL"/>
   </IDL>
</ION_DATA_OUT>
</ION_BODY>
</ION_SCRIPT>
```

The FORMAT attribute causes the ION Script variable to be evaluated differently than if the variable were simply inserted into the IDL block. This page produces the following output:

```
INT       =      123
LONG      =          123
```

Single vs. Double Quotation Marks

In IDL, strings can be enclosed in either single or double quotation marks. For example, both the following statements are valid:

```
print, "Hello World!"
print, 'Hello World!'
```

Note, however, that double quotation marks have a special meaning when followed by a number. This syntax is used to denote an octal constant. Therefore, the following statement is *not* valid:

```
print, "1 Elm Street"
```

To avoid the syntax error caused by the above statement, you can enclose your string in single quotation marks:

```
print, '1 Elm Street'
```

Keep this in mind when using ION Script variables in an <IDL> block as well. For example, suppose you have defined the following ION Script variable:

```
<VARIABLE_DECL NAME="Address" VALUE="'1 Elm Street'" TYPE="STR" />
```

Assume you used the following code to print the value of this variable in an <IDL> block:

```
print, "$Address"
```

This code would be evaluated as

```
print, "1 Elm Street"
```

Because "1 has a special meaning in IDL, the above code would cause a syntax error. Instead, you would need to enclose the variable in single quotation marks as follows:

```
print, '$Address'
```

Commenting IDL Code

IDL code inside the <IDL>...</IDL> block can contain comments. IDL comments begin with the semicolon character (;). Everything after the ; character is ignored on that line. Comments can exist on their own line, or at the end of a line of code, as in the following example:

```
<IDL>
   ; This comment is on its own line
   A = 10 ; This comment is on the same line as a command
</IDL>
```

Note

Comments cannot exist on a line that contains the line continuation character. See the following section.

Using the $ Symbol in an IDL Block

The $ symbol can have three different meanings when used in an IDL block:

- To indicate that what immediately follows the $ symbol is an ION Script variable, allowing you to use ION Script variables in IDL code.
- The IDL line-continuation character, used to continue a line of code on the next line.
- The literal $ symbol, when you want IDL to print the actual $ symbol.

Using $ to Insert ION Script Variables in IDL Code

If ION Script encounters a $ symbol in an IDL block, and the $ symbol is not preceded by the \ character, it will interpret whatever follows the $ symbol as an ION Script variable, unless the $ symbol is the last character on the line (including spaces). This allows you to use ION Script variables in your IDL code. The value of the ION Script variable will be substituted for the variable name before being passed to IDL.

Using $ as the IDL Line Continuation Character

When a line of IDL code extends beyond a certain length, it is often desirable to continue the code on the next line, thereby making your IDL code more readable. The IDL line continuation character indicates that the current statement is continued on the following line. The $ symbol must be the last character on the line for the line continuation to work. Comments (;) after the $ symbol will cause an error because the $ symbol is also used to denote an ION Script variable. Anything following the $

symbol in an ION Script page is considered a variable. The following example illustrates what is and what is not allowed when using the line continuation character:

```
<IDL>
   ; The following is a legal use of line continuation:
   A = 'This is ' + $
      'legal'
   print, A

   ; The following is *not* a legal use of line continuation
   ; because the $ character is not the last character:
   B = 'This is ' + $ ; Cannot put a comment here
      'illegal'
   print, B

   C = 'This is also ' + $
      'legal' ; A comment is allowed here because the line
              ; does not contain a $ character
   print, C
</IDL>
```

Making IDL Print the $ Symbol

If you want a $ symbol to be interpreted by IDL as a $ symbol instead of a line continuation character, you must precede the $ symbol with a backslash, \. For example, if you want IDL to print the value of the variable cost, preceded by the $ symbol, you would use the following code:

```
<ION_DATA_OUT>
   <IDL>
      cost=100
      print, "\$" + string(cost)
   </IDL>
</ION_DATA_OUT>
```

HTML Mapping

The IDL tag does not produce HTML and therefore does not support any additional attributes.

INPUT

The INPUT tag is an HTML tag used to define a form element, such as a pushbutton or a text input field. In ION_SCRIPT, the INPUT tag is used inside the ION_FORM tag to define form elements for the ION form. All INPUT types can be used in ION Script applications. For a few types, however, ION Script has custom tags with more functionality than their corresponding INPUT type:

- Use the ION_BUTTON tag instead of <INPUT TYPE="BUTTON">. Buttons created with ION_BUTTON execute an ION Script EVENT when clicked.
- Use the ION_RADIO tag instead of <INPUT TYPE="RADIO">. The ION_RADIO tag allows you to maintain the selection state of a radio button group when the page is reloaded.
- Use the ION_CHECKBOX tag instead of <INPUT TYPE="CHECKBOX">. The ION_CHECKBOX tag allows you to maintain the selection state of a checkbox when the page is reloaded.

Note

Not all attributes of the INPUT tag are documented here. Only the most commonly-used attributes are listed. Refer to an HTML reference, such as the one at http://www.htmlhelp.com, for a complete list of all INPUT tag attributes.

Syntax

<INPUT NAME="*name*"

TYPE= {"BUTTON" |
"CHECKBOX" [CHECKED] |
"FILE" |
"HIDDEN" |
"IMAGE" [ALT="*text*"] [SRC="*url*"] [ALIGN={top | middle | bottom | left | right}]|
"PASSWORD" [MAXLENGTH="*characters*"] [SIZE="*characters*"] |
"RADIO" [CHECKED] |
"RESET" |
"SUBMIT" |
"TEXT" [MAXLENGTH="*characters*"] [SIZE="*characters*"] }

[ONBLUR=*script*] [ONCHANGE=*script*] [ONFOCUS=*script*]
[TABINDEX="*position*"] [VALUE="*value*"] >

Attributes

Note

To use ION Script variables in an INPUT attribute, you must insert the variable using ION_VARIABLE. See "Variable Substitution in Attribute Values" on page 117.

ALIGN

For IMAGE fields, ALIGN specifies either the position of the image in relation to the left and right borders of the browser, or the alignment of text in relation to the image:

- LEFT - The image is aligned with the left border of the browser.
- RIGHT - The image is aligned with the right border of the browser.
- TOP - Text to the left and right of the image is aligned with the top edge of the image.
- BOTTOM - Text to the left and right of the image is aligned with the bottom edge of the image.
- MIDDLE - Text to the left and right of the image is aligned with the vertical midpoint of the image.

ALT

For IMAGE fields, ALT specifies the text to be displayed in browsers that do not support images, or in browsers that have been set to not display in-line images.

CHECKED

Specifies whether a checkbox or radio button is checked or unchecked. This attribute is not set to a value—it is either included or not included. If included, the button state is checked.

MAXLENGTH

The maximum number of characters that can be entered in a TEXT or PASSWORD form field.

NAME

The name of the form field. This name is used to refer to the value of the form field using the format $Form.*NAME*. The NAME you specify for form fields cannot begin with ION_ _.

ONBLUR

The script to execute when the form field loses focus. Focus is lost when the user clicks the mouse outside of the form field, or presses the Tab key.

ONCHANGE

The script to execute when the value in the form field changes. This event occurs when the form field loses focus, but only if the value has changed since the last time the field had focus.

ONFOCUS

The script to execute when the form field receives focus. Focus is obtained when the user clicks in or tabs into the form field.

SIZE

The length of a TEXT or PASSWORD field, in characters.

SRC

The URL of the image to use for IMAGE form fields.

TYPE

The type of form field. Valid values include BUTTON, CHECKBOX, FILE, HIDDEN, IMAGE, PASSWORD, RADIO, RESET, SUBMIT, and TEXT.

TABINDEX

The position of the form field in the Tabbing order. The Tabbing order is the order in which fields receive focus when the Tab key is pressed.

VALUE

The value assigned to the form field, which can be accessed using $Form.*NAME*. If this value is numeric, and you want to use it in an expression, you must declare a new INT or DOUBLE variable, assign it the value $Form.*NAME*, and use the new variable in the expression instead of $Form.*NAME*. See "Using $Form Variables in Numeric Expressions" on page 54.

This attribute applies to the different form elements as follows:

- For TEXT fields, VALUE specifies the text that initially appears in the field.
- For BUTTON fields, VALUE specifies the button label.

- For SUBMIT and RESET buttons, VALUE specifies the button label. If VALUE is not specified, “Submit Query” and “Reset” are the default button labels.
- For CHECKBOX and RADIO buttons, VALUE does *not* specify whether the button is checked or unchecked (this is specified with the CHECKED attribute). Instead, VALUE simply assigns a value to the checkbox or radio button.
- For HIDDEN fields, VALUE specifies the value that is passed to the server upon form submission.

ION_BODY

The ION_BODY tag pair delimits the body section of an ION Script document. The body is the part of the ION Script document that is converted into HTML for display in a Web browser. All valid HTML tags are allowed in the body (that is, between the <ION_BODY> and </ION_BODY> tags).

Note that it is not necessary to include the <HTML> or <BODY> tags in an ION Script page. ION Script inserts these elements automatically.

Syntax

<ION_BODY
[BACKGROUND="*image url*"]
[BGCOLOR="*rgb value or color name*"]
[LINK="*rgb value or color name*"]
[TEXT="*rgb value or color name*"]
[VLINK="*rgb value or color name*"]
[ALINK="*rgb value or color name*"]
[ONLOAD="*script*"]
[ONUNLOAD="*script*"]
[*Any other <BODY> attributes*] >

ION and HTML content

</ION_BODY>

Note

HTML allows you to specify <BODY> attributes without using quotation marks. ION Script is based on XML, which requires the use of quotation marks. Make sure you enclose all <ION_BODY> attributes inside double quotation marks.

Attributes

Note

Several of the following attributes can be specified either as an RGB triplet, using the general syntax *ATTRIBUTE*="*#RRGGBB*", or as a valid color name, using the syntax *ATTRIBUTE*="*color name*". See http://www.htmlhelp.com/cgi-bin/color.cgi for a list of RGB values and color names. Using RGB values instead of color names is recommended because not all browsers support color names.

BACKGROUND

The URL of the background image to use for the document.

BGCOLOR

The background color of the page. Can be specified either as BGCOLOR="*#RRGGBB*" or BGCOLOR="*color name*".

TEXT

Text color. Can be specified either as TEXT="*#RRGGBB*" or TEXT="*color name*".

LINK

Color of hyperlinks. Can be specified either as LINK="*#RRGGBB*" or LINK="*color name*".

VLINK

Color of visited hyperlinks. Can be specified either as VLINK="*#RRGGBB*" or VLINK="*color name*".

ALINK

Color of active hyperlinks. Can be specified either as ALINK="*#RRGGBB*" or ALINK="*color name*".

ONLOAD

The script to execute when the page is loaded. See “Using JavaScript and VBScript” on page 90 for information on specifying this attribute.

ONUNLOAD

Script to execute when document has been exited. See “Using JavaScript and VBScript” on page 90 for information on specifying this attribute.

HTML Mapping

The ION_BODY tag is converted to the HTML <BODY> tag. All attributes of the <BODY> tag can be included in the ION_BODY tag. Note that the attributes listed above are simply attributes of the <BODY> tag, and are listed here for convenience.

ION_BUTTON

An ION_BUTTON is similar to an HTML button created with <INPUT TYPE="SUBMIT">. The ION_BUTTON is a special input type that can have an ION Script event assigned to it. When the button is pressed, or if the image is clicked for IMAGE type buttons, the event is processed. If the ION_BUTTON is contained in a form, only those inputs local to the form are passed on to the new page.

An ION_BUTTON can be either a standard HTML button or an image. If the TYPE is BUTTON, a standard button is used and the label comes from the LABEL attribute. IMAGE buttons get their visual representation from the URL specified in the SRC attribute.

When TYPE is set to IMAGE, you can retrieve the location of the mouse when clicked on the image by using the $Mouse.x and $Mouse.y system variables.

Syntax

<ION_BUTTON
TYPE="BUTTON"
EVENT="*eventName*"
[LABEL="*label*"]
[METHOD={"GET" | "POST"}]
[<*INPUT TYPE="SUBMIT"*> *attributes*] />

or

<ION_BUTTON
TYPE="IMAGE"
EVENT="*eventName*"
SRC="*url*"
[BORDER="*width*"]
[METHOD={"GET" | "POST"}]
[<*INPUT TYPE="IMAGE"*> *attributes*] />

Attributes

BORDER

The width of the border drawn around the image. A value of 0 causes the image to be displayed without a border. The default value is 1, but this default can be changed on the “Images” tab of the ION Script Configuration utility (Windows), or in the “Images” section of the `.ionsrc` file (UNIX). This attribute is used only when TYPE is set to IMAGE. This attribute can contain a variable.

Note

<ION_BUTTON TYPE="IMAGE"> creates <INPUT TYPE="IMAGE">. Note that not all browsers support borders on <INPUT TYPE="IMAGE">.

EVENT

The name of the EVENT to execute when the user clicks the button. This name must be one of the names defined in the NAME attribute of an EVENT_DECL tag. This attribute can contain a variable.

LABEL

The text that appears on the button face. This attribute can contain a variable.

METHOD

The method used to submit form data. Valid values include:

- GET — Passes information via the URL. The amount of information that can be passed using the GET method is limited by server and browser URL length limitations. An advantage of the GET method, however, is that the user can bookmark the URL, and the browser's "Reload" button works seamlessly. This is the default method.
- POST — Form input is submitted as an HTTP POST request. POST does not impose the length restrictions imposed by GET. Therefore, the POST method can be used to submit forms containing amounts of data that would be too large to submit in a URL using the GET method. A disadvantage of the POST method is that the resulting page cannot be bookmarked, and if the user clicks the browser's "Reload" button, a "Repost form data?" dialog will appear.

If the ION_BUTTON is already contained in an ION_FORM, the METHOD attribute of ION_BUTTON is ignored and the METHOD attribute of ION_FORM is used instead.

Refer to an HTML/HTTP reference for more information on GET and POST.

SRC

The URL of the image to use for buttons of TYPE="IMAGE". See "Specifying URLs" on page 70. This attribute can contain a variable.

TYPE

The button type. Specify BUTTON to create a standard HTML Submit button (created with the HTML tag <INPUT TYPE="SUBMIT">). Specify IMAGE to create an image that, when clicked, executes an event.

HTML Mapping

The ION_BUTTON tag is converted to the HTML <INPUT> tag. The TYPE attribute of the ION_BUTTON tag becomes the TYPE attribute of the <INPUT> tag. All attributes of the <INPUT> tag can be included in the ION_BUTTON tag, except for NAME and VALUE.

If the ION_BUTTON is not contained in an ION_FORM, a <FORM> tag is also created, and METHOD attribute of ION_BUTTON becomes the METHOD of the <FORM> tag. If the ION_BUTTON is already contained in an ION_FORM, the METHOD attribute of ION_BUTTON is ignored and the METHOD attribute of ION_FORM becomes the METHOD of the <FORM> tag.

If TYPE="BUTTON"

If specified as

```
<ION_BUTTON TYPE="BUTTON" EVENT="event" LABEL="label"/>
```

then the ION_BUTTON tag is converted to the following HTML:

```
<INPUT TYPE="SUBMIT" NAME="event" VALUE="label">
```

The EVENT attribute becomes the NAME attribute of the <INPUT> tag, and the LABEL attribute becomes the VALUE attribute of the <INPUT> tag.

If TYPE="IMAGE"

If specified as

```
<ION_BUTTON TYPE="IMAGE" SRC="url" EVENT="event" BORDER="1"/>
```

then the ION_BUTTON tag is converted to the following HTML:

```
<INPUT TYPE="IMAGE" NAME="event" BORDER="1" SRC="url">
```

The EVENT attribute becomes the NAME attribute of the <INPUT> tag.

ION_CHECKBOX

An ION_CHECKBOX is similar to an HTML checkbox created with <INPUT TYPE="CHECKBOX">. ION_CHECKBOX allows you to maintain the selection state of a checkbox when the page is reloaded.

Syntax

<ION_CHECKBOX
NAME="*name*"
VALUE="*value*"
[<*INPUT TYPE="CHECKBOX"*> *attributes*] />

Attributes

NAME

The name of the checkbox. This name is used to refer to the value of the checkbox using the format $Form.*NAME*. This attribute can contain a variable. The NAME you specify for form fields cannot begin with ION_ _.

VALUE

The value assigned to the checkbox, which can be accessed using $Form.*NAME*. If this value is numeric, and you want to use it in an expression, you must declare a new INT or DOUBLE variable, assign it the value $Form.*NAME*, and use the new variable in the expression instead of $Form.*NAME*. See “Using $Form Variables in Numeric Expressions” on page 54. This attribute can contain a variable.

Note

All attributes of the HTML <INPUT TYPE="CHECKBOX"> are also available to ION_CHECKBOX. See “INPUT” on page 129.

HTML Mapping

The ION_CHECKBOX tag is converted to the HTML <INPUT TYPE="checkbox"> tag. The NAME and VALUE attributes of the ION_CHECKBOX tag become the NAME and VALUE attributes of the <INPUT> tag. All attributes of the HTML <INPUT> tag are also available to ION_CHECKBOX.

Example

In this example, named `checkbox.ion`, we create three checkboxes, and a button with an EVENT that reloads the page. When the user clicks the button, the page is reloaded, and the state of each checkbox is written to the page. When the page reloads, the checked boxes remain checked. Note that this example could also be created using checkboxes created with the INPUT tag instead of ION_CHECKBOX, but when the page reloads, the state of the checkboxes would be lost.

```
<ION_SCRIPT>
<ION_HEADER>
<EVENTS>
   <EVENT_DECL NAME="RELOAD" ACTION="ion://checkbox.ion" />
</EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
      <ION_CHECKBOX NAME="Box1" VALUE="1"/> Box 1 <BR>
      <ION_CHECKBOX NAME="Box2" VALUE="2"/> Box 2 <BR>
      <ION_CHECKBOX NAME="Box3" VALUE="3"/> Box 3 <BR>
      <ION_BUTTON EVENT="RELOAD" LABEL="SUBMIT" TYPE="BUTTON"/>
   </ION_FORM><BR>
   Box 1 was
   <ION_IF EXPR="$Form.Box1 ISTYPE 'UNDEF' ">
      unchecked<BR>
   <ION_ELSE/>
      checked<BR>
   </ION_IF>
   Box 2 was
   <ION_IF EXPR="$Form.Box2 ISTYPE 'UNDEF' ">
      unchecked<BR>
   <ION_ELSE/>
      checked<BR>
   </ION_IF>
   Box 3 was
   <ION_IF EXPR="$Form.Box3 ISTYPE 'UNDEF' ">
      unchecked
   <ION_ELSE/>
      checked
   </ION_IF>
</ION_BODY>
</ION_SCRIPT>
```

Note that when a checkbox is unchecked, the $Form variable associated with that checkbox is undefined. This allows us to use the ISTYPE operator on the $Form variable to determine whether a checkbox is checked.

To cause an ION_CHECKBOX to be initially selected when the page is loaded:

- Encode one or more name/value pairs in the URL that loads the page. If the value you pass in for a checkbox matches the value defined in the VALUE attribute of the ION_CHECKBOX tag, the checkbox will be checked. The following URL specifies that Box3 (which, in the above example, was defined with VALUE="3") be initially checked:

  ```
  http://myserver/cgi-bin/ion-p?page=checkbox.ion&Box3=3
  ```

- If the page is loaded as the action of a form, use a hidden form field to pass a name/value pair for each checkbox you want checked:

  ```
  <FORM ACTION="http://myserver/cgi-bin/ion-p">
     <INPUT NAME="page" VALUE="checkbox.ion"  TYPE="HIDDEN">
     <INPUT NAME="Box3" VALUE="3" TYPE="HIDDEN">
     <INPUT TYPE="SUBMIT">
  </FORM>
  ```

ION_DATA_OUT

The ION_DATA_OUT tag pair is used to insert text from an IDL session into an ION Script document. The final text can come from either the standard IDL output stream or it can be the textual representation of an IDL variable. An <IDL> block inside the ION_DATA_OUT tag contains the IDL code that creates the text or data to be output.

Syntax

<ION_DATA_OUT
[ASTEXT={"TRUE" | "FALSE"}]
[DEBUG={"TRUE" | "FALSE"}]
[METHOD={"GET" | "POST"}]
[PRE={"TRUE" | "FALSE"}]
[SERVER="*url*"] >

<IDL> *IDL code* (can contain $*variable*) </IDL>

</ION_DATA_OUT>

Attributes

ASTEXT

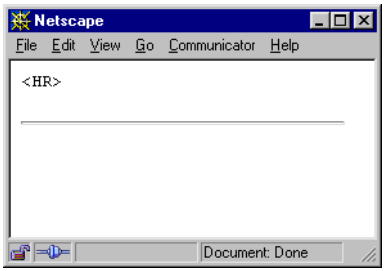
If your IDL output contains HTML tags or text that contains characters used in HTML, and you want these characters displayed as text (such as a code listing) rather than being rendered as HTML, set this attribute to TRUE. When ASTEXT is set to TRUE, ION Script converts special characters in text output from IDL, such as < and &, to character entities so that they will not be interpreted as HTML by the browser. Setting this attribute to FALSE prevents ION Script from converting special characters to character entities. The default is FALSE.

The following example illustrates the difference between ASTEXT="TRUE" and ASTEXT="FALSE":

```
<ION_SCRIPT>
<ION_BODY>
   <ION_DATA_OUT PRE="TRUE" ASTEXT="TRUE">
      <IDL>
         print, '<HR>'
      </IDL>
   </ION_DATA_OUT>
   <ION_DATA_OUT PRE="TRUE" ASTEXT="FALSE">
      <IDL>
         print, '<HR>'
```

```
        </IDL>
    </ION_DATA_OUT>
</ION_BODY>
</ION_SCRIPT>
```

The following figure shows the result of the above code:



DEBUG

Setting this attribute to TRUE causes the following:

- ION Script generates a text file that can be used to help debug your ION Script application. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL. The file is given the name specified by the “ION Debug Filename” setting on the “Debug” tab of the ION Script Configuration Utility (Windows) or in the “Debug” section of the `.ionsrc` file (UNIX). For example, if the “ION Debug Filename” setting is “ion-out%d.txt”, then a file called `ion-out<PID>.txt` is created, where `<PID>` is the unique process ID for the instance of `ion-i` that generated the output. The file is written to the directory specified by the “ION Debug Location” setting on the “Debug” tab of the ION Script Configuration Utility (Windows) or in the “Debug” section of the `.ionsrc` file (UNIX). In order for this to work, the directory specified by the “ION Debug Location” must be writable. See “Debug” on page 24 for more information.
- The IDL system variable !QUIET is set to zero, which causes all IDL messages to appear in the generated Web page.

The default value for DEBUG is FALSE.

METHOD

The method is used to submit the IDL code contained in the <IDL> block to ion-i. Valid values include:

- GET — Code in the <IDL> block is submitted via the URL. The amount of information that can be passed using the GET method is limited by server and browser URL length limitations.
- POST — Code in the <IDL> block is submitted as an HTTP POST request. POST does not impose the length restrictions imposed by GET. Therefore, the POST method can be used to submit <IDL> blocks containing amounts of code that would be too large to submit in a URL using the GET method.

Refer to an HTML/HTTP reference for more information on GET and POST.

PRE

Determines how the IDL output is treated by the ION Script parser:

- If set to TRUE, the text inserted by ION_DATA_OUT is formatted exactly as output by IDL. Text is rendered in a fixed-width font, line breaks are preserved, whitespace (including spaces and tabs) is preserved, and word wrapping will not occur. This is equivalent to using the HTML <PRE>...</PRE> block to surround text that you want formatted as typed.
- If set to FALSE, line breaks and whitespace in IDL output will be ignored, and the output will be displayed in the browser's default font (or in the font specified by an HTML tag, if used).

The default value for PRE is TRUE, but this default can be changed on the Format tab of the configuration utility.

SERVER

Note

This attribute is deprecated. If provided, it will be ignored.

The full path to the image server, `ion-i`. This must be a valid, absolute URL using the http:// protocol. This attribute, which is also an attribute of <ION_IMAGE> and <ION_OBJECT>, can be used for load control by specifying different servers for different tasks (provided you have a copy of `ion-i` on each server, and that you have the proper license). If this attribute is not specified, ION Script will use the server specified during installation. The default value for this attribute can be changed using the Image Server URL field on the "URLs" tab of the configuration utility

(Windows) or the “URLs” section of the `.ionsrc` file (UNIX). This attribute can contain a variable.

HTML Mapping

The ION_DATA_OUT tag out creates HTML content based on the textual output from IDL. The PRE attribute determines whether the textual output from IDL is contained in an HTML <PRE>...</PRE> block. PRE="TRUE" causes the IDL output to be inserted in an HTML <PRE>...</PRE> block, and PRE="FALSE" causes the IDL output to be inserted directly into the HTML stream.

For example, the ION Script code

```
<ION_DATA_OUT PRE="TRUE">
   <IDL>
      print, 'hello'
   </IDL>
</ION_DATA_OUT>
```

produces the following HTML:

```
<PRE>hello</PRE>
```

If PRE="FALSE" is used, then the text “hello” is simply inserted into the HTML stream.

Examples

Example 1

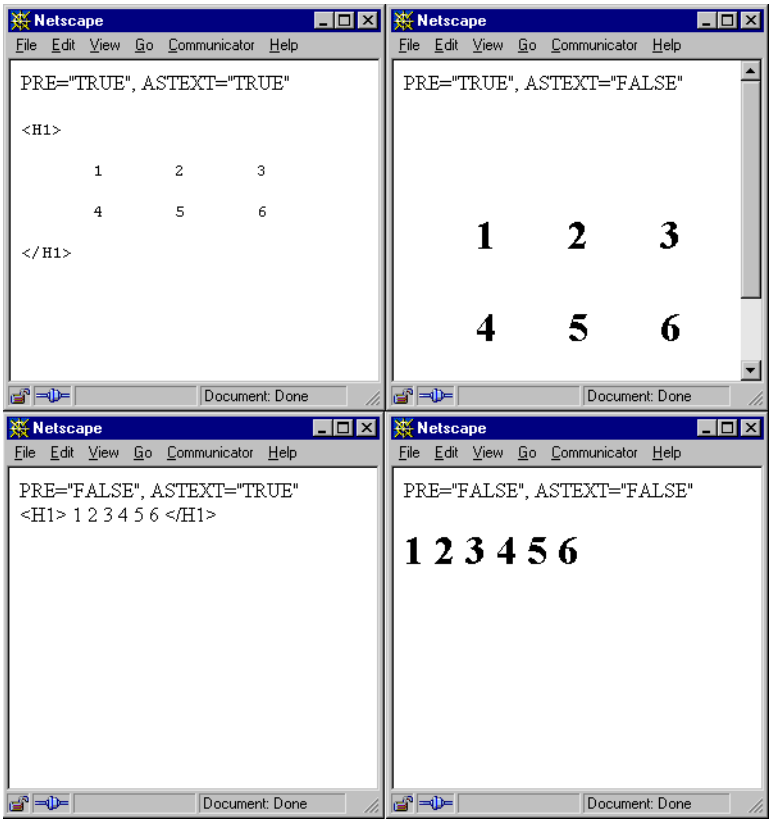
This example shows how the ASTEXT and PRE attributes can be used in four possible combinations to achieve different output.

1. Create the following ION Script file. Set the value of the ASTEXT and PRE attributes to TRUE and then to FALSE:

```
<ION_SCRIPT>
<ION_BODY>
   PRE="TRUE", ASTEXT="FALSE"<BR>
   <ION_DATA_OUT PRE="TRUE" ASTEXT="FALSE">
      <IDL>
         A = [1,2,3]
         B = [4,5,6]
         print, '<H1>'
         print, A
         print, B
         print, '</H1>'
      </IDL>
   </ION_DATA_OUT>
```

```
</ION_BODY>
</ION_SCRIPT>
```

The following figure shows the resulting output in the browser for the four combinations of ASTEXT and PRE:



Example 2

In this example, we will use IDL to dynamically generate HTML. The first page of this application allows the user to choose which data file to display as an HTML table. The EVENT page passes the path and filename to an IDL procedure that generates an HTML using data from the specified file. Note that unlike Example 1, we must call an IDL `.pro` file because we use an IDL FOR loop, which cannot be used at the command line, and therefore cannot be used in an <IDL> block.

Note

This example can be run by running the file `pickfile.ion` in the examples directory.

1. In the `ION61/ion_script/examples/data` directory, there are two ASCII text files named `mydata1.dat` and `mydata2.dat` containing a rectangular matrix of data such as the following:

```
12 53 42 34 55 66 74 90
31 10 12 23 19 18 15 10
90 76 50 34 78 86 28 33
23 53 87 42 28 64 97 18
```

2. In the `ION61/ion_script/examples` directory is the ION Script file `pickfile.ion`, which is the starting page of the application:

```
<ION_SCRIPT>
<ION_HEADER>
<EVENTS>
   <EVENT_DECL NAME="CreateTable" ACTION="ion://createtable.ion"/>
</EVENTS>
</ION_HEADER>
<ION_BODY>
   <U><B>Choose a data file to display:</B></U><BR><BR>
   <TT><ION_LINK EVENT="CreateTable" NAME1="filename"
   VALUE1="mydata1.dat">mydata1.dat</ION_LINK></TT>
   <BR><BR>
   <TT><ION_LINK EVENT="CreateTable" NAME1="filename"
   VALUE1="mydata2.dat">mydata2.dat</ION_LINK></TT>
</ION_BODY>
</ION_SCRIPT>
```

3. In the `ION61/ion_script/examples` directory is the EVENT page `createtable.ion`. Note that the form variable $Form.filename holds the name of the data file corresponding to the clicked ION_LINK. Also note that the FILEPATH function in the <IDL> block points to the data files in the `ion_script/examples/data` directory:

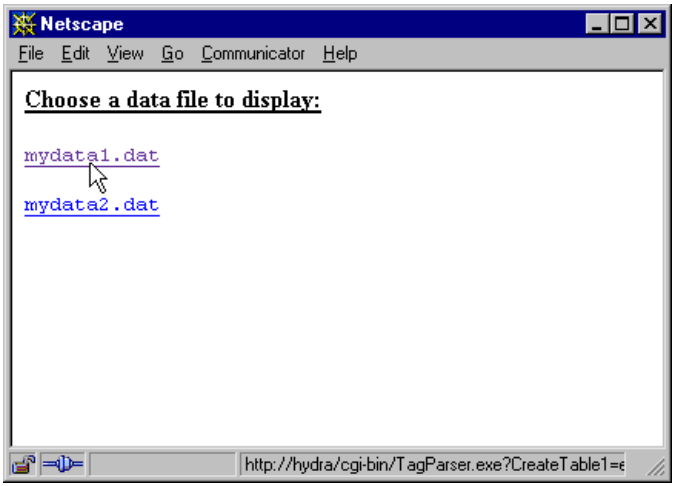
```
<ION_SCRIPT>
<ION_BODY>
   <ION_DATA_OUT PRE="FALSE">
      <IDL>
         path = FILEPATH($
            SUBDIR=['..', 'ion_script','examples', 'data'],$
            '$Form.filename')
         datafile = '$Form.filename'
         CREATETABLE, path, datafile
      </IDL>
```

```
   </ION_DATA_OUT>
</ION_BODY>
</ION_SCRIPT>
```

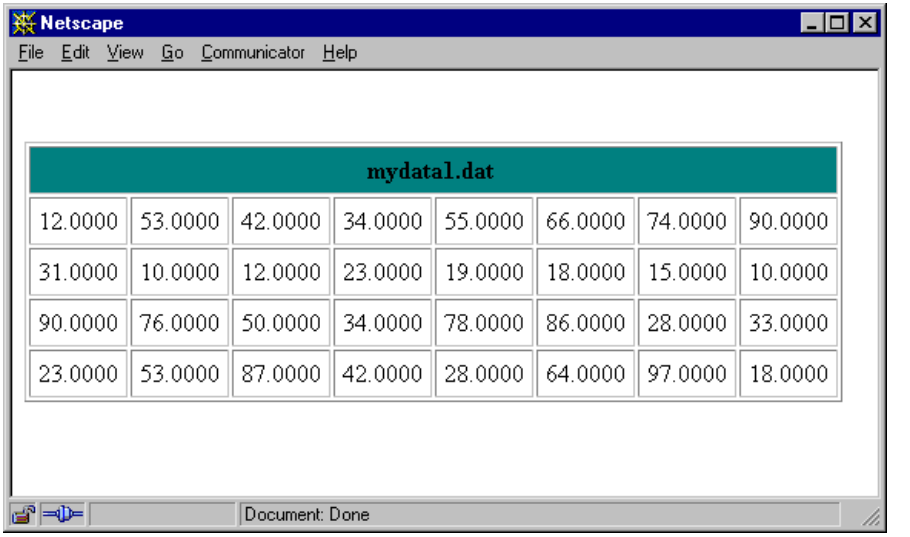
4. In the `ION61/ion_script/examples` directory is the IDL file `createtable.pro` that generates the HTML:

```
PRO CREATETABLE, path, datafile

   ; Read ascii data file:
   RESULT = READ_ASCII(path)

   ; Find dimensions:
   NUMDIMS=SIZE(RESULT.FIELD1, /N_DIMENSIONS)
   DIMS=SIZE(RESULT.FIELD1, /DIMENSIONS)
   IF (NUMDIMS EQ 1) THEN BEGIN
      COLUMNS=1
      ROWS=DIMS[0]
   ENDIF ELSE BEGIN
      COLUMNS=DIMS[0]
      ROWS=DIMS[1]
   ENDELSE
   ; Output HTML table:
   PRINT, '<BR><BR><CENTER><TABLE BORDER="1" CELLPADDING="5">'
   PRINT, '<TR><TH BGCOLOR="#008080" COLSPAN="' + $
      STRTRIM(COLUMNS,1) + '">'
   PRINT, datafile
   PRINT, '</TH></TR>'
   IF (COLUMNS NE 1) THEN BEGIN
      FOR ROW=0, ROWS-1 DO BEGIN
         PRINT, '<TR>'
         FOR COL=0, COLUMNS-1 DO BEGIN
            PRINT, '<TD WIDTH="30">'
            PRINT, RESULT.FIELD1[COL, ROW]
            PRINT, '</TD>
         ENDFOR
         PRINT, '</TR>'
      ENDFOR
      PRINT, '</TABLE></CENTER>'
   ENDIF ELSE BEGIN
      ; If only one column:
      FOR ROW=0, ROWS-1 DO BEGIN
         PRINT, '<TR><TD WIDTH="30">'
         PRINT, RESULT.FIELD1[ROW]
         PRINT, '</TD></TR>'
      ENDFOR
      PRINT, '</TABLE></CENTER>'
   ENDELSE
END
```

The following figure shows the starting page of this application, `pickfile.ion`:



When the user selects `mydata1.dat`, the following page is returned:



mydata1.dat							
12.0000	53.0000	42.0000	34.0000	55.0000	66.0000	74.0000	90.0000
31.0000	10.0000	12.0000	23.0000	19.0000	18.0000	15.0000	10.0000
90.0000	76.0000	50.0000	34.0000	78.0000	86.0000	28.0000	33.0000
23.0000	53.0000	87.0000	42.0000	28.0000	64.0000	97.0000	18.0000

ION_EVALUATE

The ION_EVALUATE tag evaluates expressions inside an ION Script document. It can simply evaluate an expression or output the result of the expression into the resulting document.

Syntax

<ION_EVALUATE
EXPR="*expression*"
[DISPLAY={"TRUE" | "FALSE"}]
[FORMAT="*format string*"] />

Attributes

EXPR

The expression to be evaluated by the parser. This attribute can contain a variable. See Chapter 3, “Variables, Expressions & Operators” for details on writing ION Script expressions.

Note that the way you declare a variable is different from the way you use ION_EVALUATE to assign a value to a variable. When declaring a variable with VARIABLE_DECL, you do *not* prefix the variable name with the $ symbol. When you assign a variable a value using ION_EVALUATE, you *do* prefix the variable name with the $ symbol, regardless of whether or not the variable has already been declared. For example:

```
<ION_EVALUATE EXPR="$Temperature = 98.6"/>
```

Warning

Assigning `$Browser.REMOTE_USER` as a string variable to the VALUE attribute can be dangerous because it is frequently an undefined system variable. When VALUE is undefined, IDL reports an error. However, it is possible that code that uses `$Browser.REMOTE_USER` could be implemented and tested without error on one system and fail if ported to another system.

DISPLAY

Determines whether or not to display the result of the expression in the outputted HTML page. The default value is FALSE.

FORMAT

Specifies the C-style printf() formatting string. The syntax for the FORMAT attribute is as follows:

FORMAT="%[*flags*][*width*][.*precision*][*type length*] *specifier*"

where

flags is one or more of the following characters:

Flag	Description
space	Precede the result with a space
-	Left-justify the result within the field if width is specified
+	Precede the result with a sign (+ or -). The default is to display only a minus sign.
#	• If specifier is o, precede result with “0” • If specifier is x, precede result with “0x” • If specifier is X, precede result with “0X” • If specifier is e, E, f, g, G, the decimal point is always present in the result, even if no digits follow the decimal • If specifier is g, G, trailing zeros will display
0	Pad the result with leading zeros (if the specified width is greater than the width of the result).

width is an integer specifying the minimum field width. If the result has fewer characters than the field width, it is padded on the left (or right, if the - flag has been specified). The padding is done with spaces unless the first character of *width* is a zero, in which case the padding is done with zeros.

precision is an integer preceded by a decimal point specifying the minimum number of digits to appear for the d, o, or x format specifiers, or the maximum number of characters to be printed for the s format specifier. It is also the precise number of digits to display after the decimal (including trailing zeros) for the e, E, f, g, or G specifiers.

type length is one of the following:

Type Length	Description
h	For d, i, o, u, x or X specifiers: value is treated as a short or unsigned short.
l (lower case L)	The value is treated as a long or unsigned long
L	The value is treated as a long double

specifier is one of the following:

ION Variable Type	Valid Specifiers
BOOL, STRING	%s
INT	%d, %i, %o, %u, %x, %X
DOUBLE	%e, %E, %f, %g, %G

Note

For more information on using format specifiers, refer to a C programming manual.

HTML Mapping

The ION_EVALUATE tag inserts the result of EXPR directly into the HTML stream if DISPLAY="TRUE". If DISPLAY="FALSE", no HTML output is produced.

ION_FORM

The ION_FORM tag pair defines a form in an ION Script page. An ION Script form is similar to an HTML form, except that the action taken when an event occurs inside the form is not limited to one choice. Instead, the source of the event determines what action is taken. For example, a form can contain multiple ION_BUTTON elements. Each button can be assigned a different EVENT that occurs when the button is clicked. Only data values within a form and variables marked as persistent are passed on to the next ION Script page.

Syntax

<ION_FORM
[METHOD={"GET" | "POST"}]
[NAME="*name*"]
[TARGET="*frame target*"]
[<*FORM*> *attributes except ACTION*] >

[<*INPUT*> *tags*]

[<*ION_CHECKBOX /*> *tags*]

[<*ION_RADIO /*> *tags*]

[*Any HTML/ION tag except <FORM>, <ION_FORM>*]

</ION_FORM>

Attributes

METHOD

The method used to submit form data. Valid values include:

- GET — Passes information via the URL. The amount of information that can be passed using the GET method is limited by server and browser URL length limitations. An advantage of the GET method, however, is that the user can bookmark the URL, and the browser's "Reload" button works seamlessly. This is the default method.
- POST — Form input is submitted as an HTTP POST request. POST does not impose the length restrictions imposed by GET. Therefore, the POST method can be used to submit forms containing amounts of data that would be too large to submit in a URL using the GET method. A disadvantage of the POST

method is that the resulting page cannot be bookmarked, and if the user clicks the browser's "Reload" button, a "Repost form data?" dialog will appear.

Refer to an HTML/HTTP reference for more information on GET and POST.

NAME

The name of the form. This name can be used to access form elements and methods through JavaScript/VBScript. For example, you could use JavaScript to perform client-side validation of form data before allowing the form to be submitted to the server, thereby saving your server from having to check form data. This attribute can contain a variable.

TARGET

The name of the window or frame in which to load the URL specified by an EVENT contained within the form when the form is submitted. This attribute can be set to either the NAME attribute of an HTML <FRAME> or <IFRAME> tag, or to one of four predefined windows: _blank, _parent, _self, or _top. See "Targeting Different Frames" on page 88 for a description of these predefined windows. See "Overview of Fixed Framesets" on page 82 for an example of how to use the TARGET attribute.

ONSUBMIT

The script to execute when the form is submitted. See "Using JavaScript and VBScript" on page 90 for information on specifying this attribute.

ONRESET

The script to execute when the form is reset. See "Using JavaScript and VBScript" on page 90 for information on specifying this attribute.

Forms Containing a Single Text Field

HTML forms that contain a single text field can be submitted by pressing the Enter key when the cursor is in the text field. If your ION_FORM contains a single text field, pressing the Enter key when the cursor is in the text field will cause an error unless an event is associated with the text field. To prevent an error, make sure you have defined an event with the same name as the NAME attribute of the text field. For example, the form on the following ION Script page can be submitted with the button or the Enter key:

page1.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="BUTTON_PRESS" ACTION="ion://page2.ion"/>
      <EVENT_DECL NAME="TEXT_ENTERED" ACTION="ion://page2.ion"/>
   </EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_FORM>
      This form can be submitted by clicking the button or <BR>
      pressing the Enter key while the cursor is in the <BR>
      text field.<BR><BR>
      <INPUT NAME="TEXT_ENTERED" TYPE="TEXT" VALUE="5">
      <ION_BUTTON EVENT="BUTTON_PRESS" TYPE="BUTTON"/>
   </ION_FORM>
</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>
<ION_BODY>
   You submitted <ION_VARIABLE NAME="$Form.TEXT_ENTERED"/>
</ION_BODY>
</ION_SCRIPT>
```

Note that the ION_BUTTON uses an event with a different name than the event associated with the text field, but the ACTION is the same. This avoids submitting two parameters with the same name.

HTML Mapping

The ION_FORM tag is converted to the HTML <FORM> tag. The value of the ION Script Parser URL, which is specified in the ION Script Configuration utility or the `.ionsrc` file, is used as the value of the <FORM> tag's ACTION attribute, and the METHOD attribute of the <FORM> tag is set to "GET". For example, if you have set the ION Script Parser URL to "http://myhost.mydomain.com/cgi-bin/ion-p", then <ION_FORM> is converted to the following HTML:

```
<FORM ACTION="http://myhost.mydomain.com/cgi-bin/ion-p"
   METHOD="GET">
```

All attributes of the HTML <FORM> tag can be used in the ION_FORM tag. Note that the attributes listed above (ACTION, METHOD, etc.) are simply attributes of the <FORM> tag, and are listed here for convenience.

Note

One exception is the MULTIPLE attribute of the HTML <FORM>'s <SELECT> tag. This tag/attribute combination is not functional with ION_FORM, which then completely disables the HTML <FORM> implementation of listbox menus that allow multiple selections.

ION_HEADER

The ION_HEADER tag pair is used to declare information, called metadata, about an ION Script document. If included, the ION_HEADER section appears before the ION_BODY section.

In addition to the ION Script metadata tags listed below, HTML tags that are valid inside the HEAD element of an HTML page can also be included inside the ION_HEADER. For example, if you wish to include a script and style sheet in the HEAD element of the Web page, include the HTML <SCRIPT> and <STYLE> tags inside the ION_HEADER.

Syntax

<ION_HEADER>

[<AUTHOR> *author* </AUTHOR>]
[<APPLICATION> *application name* </APPLICATION>]
[<COPYRIGHT>*copyright*</COPYRIGHT>]
[<DATE> *date* </DATE>]
[<LASTUPDATE>*date*</LASTUPDATE>]
[<TITLE> *title* </TITLE>]

[<EVENTS> *EVENT_DECL tags* </EVENTS>]
[<VARIABLES> *VARIABLE_DECL tags* </VARIABLES>]

[Any other HTML tags that are valid inside HEAD, such as <BASE>, <LINK>, <META>, <SCRIPT>, and <STYLE>]

</ION_HEADER>

Metadata Tags

The following tags are used to provide information about an ION Script document. Search engines use this metadata to index your page according to its content. These tags are valid only inside the <ION_HEADER>...</ION_HEADER> block.

APPLICATION

The name of the ION Script application to which this page belongs.

AUTHOR

The document author of this page.

COPYRIGHT

A copyright statement. By default, ION Script automatically inserts the © symbol before the value of this tag when you include $Document.COPYRIGHT (using ION_VARIABLE). If you do not want to use this symbol, you can change the default using the Format tab of the configuration utility.

DATE

The date the document was created.

LASTUPDATE

The date the document was last updated.

TITLE

The page title. Note that this is simply the HTML <TITLE> tag. Including a <TITLE> tag in your ION_HEADER block allows you to access the variable $Document.TITLE.

Events and Variables

ION Script event and variable declarations are made in the header section, using the EVENTS and VARIABLES tags, respectively.

HTML Mapping

The ION_HEADER tag is converted to the HTML <HEAD> tag. All attributes of the HTML <HEAD> tag can be used in the opening <ION_HEADER> tag. Additionally, any HTML tags that can be nested inside the <HEAD>...</HEAD> block can be included inside the <ION_HEADER>...</ION_HEADER> block. The ION Script metadata tags are also converted to HTML, as illustrated in the following examples:

ION Script Metadata Tag	Resulting HTML
`<AUTHOR>name</AUTHOR>`	`<META NAME="Author" CONTENT="name"/>`
`<APPLICATION>myapp </APPLICATION>`	`<META NAME="Application" CONTENT="myapp"/>`
`<COPYRIGHT>2001</COPYRIGHT>`	`<META NAME="Copyright" CONTENT="2001"/>`
`<DATE>1/1/01</DATE>`	`<META NAME="Date" CONTENT="1/1/01"/>`
`<LASTUPDATE>1/1/01 </LASTUPDATE>`	`<META NAME="LastUpdate" CONTENT="1/1/01"/>`

ION_IF, ION_ELSEIF, ION_ELSE

The ION_IF tag pair is the main flow-control tag in ION Script. It allows parts of an ION Script document to be conditionally evaluated and displayed. This feature allows you to build interactive Web-based applications.

When an ION_IF tag is encountered in a Web page, the ION Script parser evaluates the expression. If the expression evaluates to TRUE, all content up to either the closing ION_IF tag or the first internal ION_ELSEIF or ION_ELSE tag is processed. If the expression evaluates to FALSE, the parser searches for an ION_ELSEIF or ION_ELSE tag. If an ION_ELSEIF tag is encountered, its expression is evaluated and processing continues as with ION_IF. If an ION_ELSE tag is encountered, the content between the opening and closing ION_IF tags is processed.

Both ION Script and HTML content can be conditionally displayed.

Syntax

<ION_IF EXPR="*expression*">

ION or HTML content

[<ION_ELSEIF EXPR="*expression*" />

ION or HTML content

[<ION_ELSE />

ION or HTML content

</ION_IF>

Attributes

EXPR

The expression to be evaluated by the parser. This attribute can contain a variable. See Chapter 3, “Variables, Expressions & Operators” for details on writing ION Script expressions.

HTML Mapping

The ION_IF, ION_ELSEIF, and ION_ELSE tags do not directly produce any HTML output, however, any valid HTML can be contained within the <ION_IF>...</ION_IF> tag pair. The EXPR attribute of the ION_IF, ION_ELSEIF, and ION_ELSE tags is evaluated to determine which content to process.

ION_IMAGE

The ION_IMAGE tag pair inserts an IDL-generated graphic into an ION Script document. Both Direct and Object graphics images can be generated. An EVENT assigned to the ION_IMAGE makes the image an input that can be clicked on to load a new ION Script document. The location of the mouse click is passed on to the new document through the Mouse.x and Mouse.y system variables.

The IDL code that generates the image is embedded an <IDL>...</IDL> block inside the ION_IMAGE block. For a Direct Graphics image, the last image created using a Direct Graphics command is rendered to the output page. For an Object Graphics image, the last image rendered to the ION_SCRIPT_BUFFER object is output in the page.

For images with events, ION Script creates the image using the HTML <INPUT TYPE="IMAGE" NAME="*EventName*" ALT="*LABEL*">. For images without events, ION Script creates the image with the HTML <IMG NAME="IMG" ALT="*LABEL*">.

Syntax

<ION_IMAGE
TYPE={"DIRECT" | "OBJECT"}
[BORDER="*width*"]
[DEBUG={"TRUE" | "FALSE"}]
[EVENT="*eventName*"]
[HEIGHT="*height*"]
[IMG_TYPE={"PNG8" | "PNG24"| "JPEG24"}]
[LABEL="*label"*]
[SERVER="*url*"]
[TARGET="*frame target*"]
[WIDTH="*width*"]
[*<IMG> attributes (if EVENT not present) or <INPUT> attributes (if EVENT present)*]>

<IDL> *IDL code* (can contain $*variable*) </IDL>

</ION_IMAGE>

Notes About Using ION_IMAGE

Note the following when using the ION_IMAGE tag:

- The IDL code inside the <IDL>...</IDL> block can consist of the actual IDL statements that generate an image, or can simply be a call to a `.pro` file that generates an image.
- Unlike ION_DATA_OUT, ION_IMAGE does not support the POST method. This is because image data is always passed to `ion-i` via an encrypted URL. If IDL blocks in your ION_IMAGE tags are large enough to exceed the maximum allowable length of a URL, such as very large data sets or a very large amount of IDL code, then you can work around the limitation by first using ION_DATA_OUT to create an IDL `.sav` file, and then simply restoring the `.sav` file in the ION_IMAGE call. For an example of this technique, see the example application `largeData.ion` in the `examples` directory.
- The IDL code in the <IDL> block of an ION_IMAGE tag is not processed until the entire ION Script page has been parsed into HTML. Therefore, ION_IMAGE cannot be used to create data that is later used in constructing the web page. For example, you cannot use ION_IMAGE to create a `.sav` file that is restored in the <IDL> block of an ION_IMAGE or ION_DATA_OUT tag on the same page. Similarly, you cannot use ION_IMAGE to create a file that is included in the same ION Script page with ION_INCLUDE. For these cases, you would need to create the data using an ION_DATA_OUT tag, in which case the IDL code is processed as soon as it is encountered.

Attributes

BORDER

The width of the border drawn around the image. A value of 0 causes the image to be displayed without a border. The default BORDER is 1, but this default can be changed on the “Images” tab of the ION Script Configuration utility (Windows), or in the “Images” section of the `.ionsrc` file (UNIX). This attribute can contain a variable.

DEBUG

Setting this attribute to TRUE causes ION Script to generate a text file that can be used to help debug your ION Script application. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL. The file is given the name specified by the “ION Debug Filename” setting on the “Debug” tab of the ION Script Configuration utility (Windows) or in the “Debug” section of the `.ionsrc` file

(UNIX). For example, if the “ION Debug Filename” setting is “ion-out%d.txt”, then a file called `ion-out<PID>.txt` is created, where `<PID>` is the unique process ID for the instance of `ion-i` that generated the output. The file is written to the directory specified by the “ION Debug Location” setting on the “Debug” tab of the ION Script Configuration utility (Windows) or in the “Debug” section of the `.ionsrc` file (UNIX). In order for this to work, the directory specified by the “ION Debug Location” must be writable. See “Debug” on page 24 for more information. The default value for DEBUG is FALSE.

EVENT

The name of the EVENT_DECL to execute when the user clicks on the image. This name must be one of the names defined in the NAME attribute of an EVENT_DECL tag. This attribute can contain a variable.

HEIGHT

The height of the image, in pixels. This attribute can contain a variable.

IMG_TYPE

The graphics file format of the image created by the ION Script Image Engine. Valid values are:

- PNG8 — 8-bit PNG images
- PNG24 — 24-bit PNG images (lossless compression)
- JPEG24 — 24-bit JPEG images (lossy compression)

Note

GIF, JPEG, and PNG are also accepted for backwards compatibility. GIF and PNG are interpreted as PNG8, and JPEG is interpreted as JPEG24.

For IMG_TYPE="DIRECT", the default is PNG8. For IMG_TYPE="OBJECT", the default is JPEG24. These defaults can be changed using the configuration utility or the `.ionsrc` file. The value specified in this attribute overrides the default value specified in the configuration utility. For information on how to select the appropriate image format, see “Graphics in ION Script” on page 91.

LABEL

The default text displayed in place of an image for browsers that do not support images, and for browsers in which the automatic display of in-line images has been turned off. If not specified, the value specified in the “Label” field on the “Images”

tab of the configuration utility (Windows), or in the “Images” section of the `.ionsrc` file (UNIX), is used as the default.

This label is also used in some cases as the tooltip that appears when you hold the mouse cursor over the image, depending on which browser type and version you are using. Because different browsers (and different versions of each) implement alternate text and tooltips differently, it helps to know how ION Script maps the label to HTML. This label value (or the LABEL attribute of ION_IMAGE, if specified) becomes the value of the ALT attribute of the HTML <IMG> tag for an ION_IMAGE without an EVENT, or the ALT attribute of the <INPUT TYPE="IMAGE"> tag for an ION_IMAGE with an EVENT. Because some browsers handle images created with the <IMG> tag differently than those created with the <INPUT> tag, the name of the EVENT may be displayed in place of the value specified for the label if the EVENT attribute of the ION_IMAGE tag has been specified for an image.

SERVER

The full path to the image engine, `ion-i`. This must be a valid, absolute URL using the HTTP or HTTPS protocol. This attribute, which is also an attribute of <ION_DATA_OUT> and <ION_OBJECT>, can be used for load control by specifying different servers for different tasks (provided you have a copy of `ion-i` on each server, and that you have the proper license). If this attribute is not specified, ION Script will use the server specified during installation. The default value for this attribute can be changed using the configuration utility. This attribute can contain a variable.

TARGET

The name of the window or frame in which to load the URL specified by the EVENT attribute when the user clicks the image. This attribute can be set to either the NAME attribute of an HTML <FRAME> or <IFRAME> tag, or to one of four predefined windows: _blank, _parent, _self, or _top. See “Targeting Different Frames” on page 88 for a description of these predefined windows. See “Overview of Fixed Framesets” on page 82 for an example of how to use the TARGET attribute.

TYPE

Specifies whether image is generated using IDL’s Direct or Object graphics system. Set to DIRECT to use IDL Direct Graphics, or to OBJECT to use Object Graphics. This is a required attribute.

WIDTH

The width of the image, in pixels. This attribute can contain a variable.

HTML Mapping

The ION_IMAGE tag is converted to HTML as either an <IMG> tag, or a <FORM> tag with a nested <INPUT> tag, depending on whether the EVENT attribute is present.

If EVENT is Not Present

If specified as

```
<ION_IMAGE TYPE="type"/>
```

then the following HTML is produced:

```
<IMG WIDTH="width" HEIGHT="height" BORDER="1"    SRC="url">
```

where `width` and `height` are defined by the “Image Width” and “Image Height” values specified in the configuration utility or `.ionsrc` file, and `url` is made up of the “ION Image Server URL” and the data returned by the <IDL> block. In this case, all attributes of the HTML <IMG> tag can be used in the <ION_IMAGE> tag.

If EVENT is Present

If specified as

```
<ION_IMAGE EVENT="event" TYPE="type"/>
```

then the following HTML is produced:

```
<FORM ACTION="ION Script Parser URL value" METHOD="GET">
   <INPUT TYPE="IMAGE" NAME="event" BORDER="1" SRC="url">
</FORM>
```

where `url` is made up of the ION Image Server URL and the data returned by the <IDL> block. In this case, all attributes of the HTML <INPUT TYPE="IMAGE"> tag can be used in the <ION_IMAGE> tag.

Example

In this simple example, we create an image that, when clicked, takes the user to another ION Script page that reports the x and y location of the mouse cursor:

`image.ion`:

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL ACTION="ion://clickloc.ion" NAME="CLICKLOC"/>
   </EVENTS>
</ION_HEADER>
```

```
<ION_BODY>
   <ION_IMAGE EVENT="CLICKLOC" IMG_TYPE="PNG8" TYPE="DIRECT">
      <IDL>
         show3, dist(30)
      </IDL>
   </ION_IMAGE>
</ION_BODY>
</ION_SCRIPT>
```

clickloc.ion:

```
<ION_SCRIPT>
<ION_BODY>
   You clicked at
   x=<ION_VARIABLE NAME="$Mouse.x"/>,
   y=<ION_VARIABLE NAME="$Mouse.y"/>
</ION_BODY>
</ION_SCRIPT>
```

ION_INCLUDE

The ION_INCLUDE tag inserts the contents of another ION Script, text, or HTML file into the current ION Script document. These file types are handled by ION Script as follows:

- ION Script files included with ION_INCLUDE are processed by the ION Script parser, and included as HTML in the final document.
- Text files are processed to translate special characters used in HTML, such as > and <, into *character entities* that can be recognized by any Web browser. For example, the > symbol is converted to >. When rendered by a Web browser, > will appear as >. This allows all characters in the text file to be rendered as they appear in the text file instead of being interpreted as HTML.
- HTML files are included unchanged.

The file extension of the included file determines whether ION Script treats the file as text, ION Script, or HTML. The ION Extensions field on the Files tab of the configuration utility specifies extensions that are to be considered ION Script files, and the Text Extensions field specifies extensions that are to be considered text files.

Note

ION Script does not support HTTP proxies. Therefore, ION_INCLUDE can only be used to include files that are accessible without a proxy. If your system resides inside a firewall that uses a proxy for accessing Web sites outside the firewall, you cannot include a Web page that resides outside the firewall.

Syntax

```
<ION_INCLUDE
        SRC="url"
        [ASTEXT={"TRUE" | "FALSE"}]
        [PRE={"TRUE" | "FALSE"}] />
```

Attributes

ASTEXT

If the file to be included contains HTML tags or text that contains characters used in HTML, and you want these characters displayed as text (such as a code listing) rather than being rendered as HTML, set this attribute to TRUE. By default, ION Script converts special characters such as < and >, to character entities so that they will not

be interpreted as HTML by the browser. Setting this attribute to TRUE prevents ION Script from converting special characters to character entities. See “Example 1” on page 144 for an example of the effect of the ASTEXT attribute.

Note that ASTEXT has no effect on included ION Script files.

PRE

Determines how the included file is treated by the ION Script parser:

- If set to TRUE, the included file is rendered in a fixed-width font, line breaks are preserved, whitespace (including spaces and tabs) is preserved, and word wrapping will not occur. This is equivalent to using the HTML <PRE>...</PRE> block to surround text that you want formatted as typed.
- If set to FALSE, line breaks and whitespace in the file will be ignored, and the file will be displayed in the browser’s default font (or in the font specified by an HTML tag, if used).

The default value is TRUE, but this default can be changed on the “Format” tab of the configuration utility (Windows) or the “Format” section of the `.ionsrc` file (UNIX). See “Use <PRE> with Text Files” on page 23. For an example illustrating the effect of the PRE attribute, see “Example 1” on page 144.

SRC

The URL of the file to include. See “Specifying URLs” on page 70. The type of file to be included is identified by its extension. This attribute can contain a variable. By default, the following file extensions are recognized by the ION Script parser:

File Type	Extension
Text	.txt
ION Script	.ion

The configuration utility or `.ionsrc` file can be used to define additional extensions for each file type. See “Text Extensions” on page 21.

HTML Mapping

The ION_INCLUDE tag is not converted to HTML, and therefore does not accept any other attributes.

Examples

For these examples, assume we have the same file saved with three different filenames: `includefile.txt`, `includefile.html`, and `includefile.ion`. The file contains the following text:

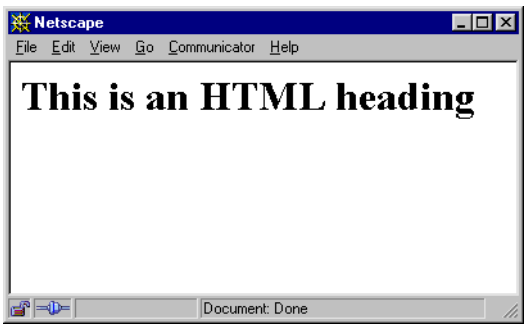
```
<ION_SCRIPT>
<ION_BODY>
   <H1>This is an HTML heading</H1>
   <ION_EVALUATE EXPR="5+5" DISPLAY="TRUE"/>
</ION_BODY>
</ION_SCRIPT>
```

Example 1

Include the file `includefile.html` as pure HTML:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_INCLUDE SRC="file://includefile.html" />
</ION_BODY>
</ION_SCRIPT>
```

This results in the following page:

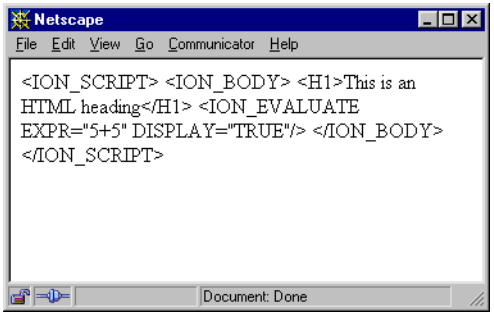


Example 2

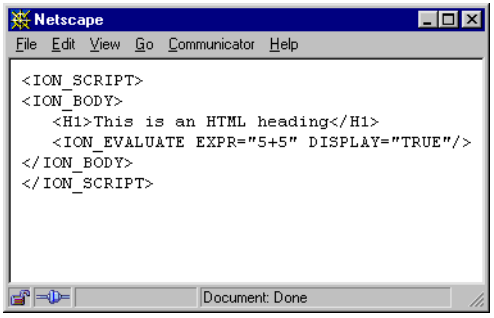
Include the HTML file as a text file, so that the HTML is not rendered as HTML by the browser, and ION Script tags are not evaluated by the ION Script parser.This can be accomplished by adding ASTEXT="TRUE" to the ION_INCLUDE tag:

```
<ION_SCRIPT>
<ION_BODY>
  <ION_INCLUDE SRC="file://includefile.html" ASTEXT="TRUE"/>
</ION_BODY>
</ION_SCRIPT>
```

This results in the following page:



If we also use PRE="TRUE" in our ION_INCLUDE tag, we get the following page:

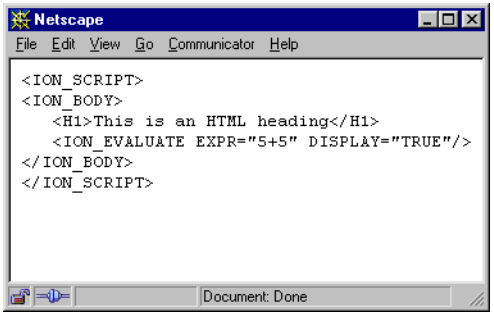


Example 3

Include the file `includefile.txt`.

```
<ION_SCRIPT>
<ION_BODY>
   <ION_INCLUDE SRC="file://includefile.txt" />
</ION_BODY>
</ION_SCRIPT>
```

Assuming `.txt` is listed as one of the Text Extensions in the configuration utility or `.ionsrc` file, the file is automatically treated as a text file. If the configuration option "Use <PRE> with Text Files" is turned on, this results in the following page. Note that this is the same as including the HTML file with ASTEXT="TRUE" and PRE="TRUE":

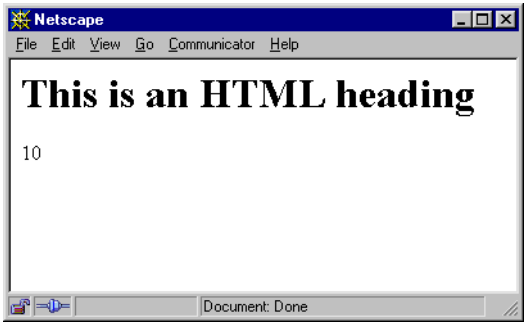


Example 4

Include the file `includefile.ion`:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_INCLUDE SRC="ion://includefile.ion"/>
</ION_BODY>
</ION_SCRIPT>
```

Assuming `.ion` is listed as one of the Text Extensions on the “Files” tab of the ION Script Configuration utility (Windows) or in the “Files” section of the `.ionsrc` file (UNIX), the file is automatically treated as an ION Script page:



ION_LINK

The ION_LINK tag is used to create a text hyperlink to another `.ion` page. Unlike an HTML hyperlink, the ION_LINK allows you to pass all persistent variables to the destination page. ION_LINK also allows you to assign a value to the link, and pass that value to the destination page, where it can be accessed as a form variable.

Syntax

<ION_LINK
EVENT="*eventName*"
[TARGET="*frame target*"]
[NAME1="*name*" VALUE1="*value*"]
. . .
[NAME*n*="*name*" VALUE*n*="*value*"] >

Link Text (can be HTML)

</ION_LINK>

Attributes

EVENT

The name of the EVENT_DECL to execute when the user clicks on the link. This name must be one of the names defined in the NAME attribute of an EVENT_DECL tag. This attribute can contain a variable.

NAME*n*

The name of the value to pass to the destination page. The first NAME*n* must be NAME1, and subsequent names must be consecutively numbered. NAME1 corresponds to the value passed in VALUE1. As with form fields, this name becomes a $Form variable named $Form.*NAME*. This attribute can contain a variable.

TARGET

The name of the window or frame in which to load the URL specified by the EVENT attribute when the link is clicked. This attribute can be set to either the NAME attribute of an HTML <FRAME> or <IFRAME> tag, or to one of four predefined windows: _blank, _parent, _self, or _top. See “Targeting Different Frames” on page 88 for a description of these predefined windows. See “Overview of Fixed Framesets” on page 82 for an example of how to use the TARGET attribute.

VALUE*n*

A value that is passed to the destination page. The first VALUE*n* must be VALUE1, and subsequent values must be consecutively numbered. VALUE1 corresponds to the value named NAME1. This value can be accessed on the destination page as $Form.*NAME*, where *NAME* is the NAME*n* attribute corresponding to this value. This attribute can contain a variable.

HTML Mapping

The ION_LINK tag is converted to the HTML <A> tag. EVENT, TARGET, and VALUE*n* are used in the construction of the URL specified as the value of the HREF attribute of the <A> tag. Any HTML nested inside the <ION_LINK>...</ION_LINK> block is passed to the browser unchanged.

Example

Create two links, and pass the value of the clicked link to the destination page. By giving each link the same name, we can use these links in the same way we use radio buttons. Note that the value of the clicked link is accessed using the $Form variable:

page1.ion:

```
<ION_SCRIPT>
<ION_HEADER>
<EVENTS>
   <EVENT_DECL NAME="PAGE2" ACTION="ion://page2.ion"/>
</EVENTS>
</ION_HEADER>
<ION_BODY>
   <ION_LINK EVENT="PAGE2" NAME1="Link" VALUE1="One">Link 1
   </ION_LINK><BR>
   <ION_LINK EVENT="PAGE2" NAME1="Link" VALUE1="Two">Link 2
   </ION_LINK>
</ION_BODY>
</ION_SCRIPT>
```

page2.ion:

```
<ION_SCRIPT>
<ION_BODY>
   <ION_IF EXPR="$Form.Link EQ 'One' ">
      <B>You selected link 1</B>
   <ION_ELSE/>
      <B>You selected link 2</B>
   </ION_IF>
</ION_BODY>
</ION_SCRIPT>
```

For another example of how the VALUE attribute of an ION_LINK can be used, see “Example 2” on page 145.

ION_OBJECT

The ION_OBJECT tag pair is used to insert content such as VRML, MPEG, and WAV files into an ION Script application. IDL is capable of creating output in a variety of file formats other than the image formats handled by ION_IMAGE and text handled by ION_DATA_OUT. The ION_IMAGE tag allows the ION Script developer to insert any IDL-generated data that can be inserted into a Web page with either the <OBJECT> tag or <EMBED> tag.

Implementation of the <OBJECT> and <EMBED> tags differs significantly from browser to browser. Therefore, it is usually necessary to write ION_OBJECT code differently for each browser supported by your application. It is important to understand how each browser you intend to support handles the MIME types you are including in your ION Script application. In addition, plug-ins are often used to render data contained in <OBJECT> and <EMBED> tags. Each plug-in has its own coding requirements, so make sure you know which information is required by the plug-ins you are supporting.

The <OBJECT> tag was originally implemented by Microsoft to support their ActiveX applets. In a similar manner, Netscape initially supported the alternative <EMBED> and <APPLET> tags for inclusion objects and later provided support for the <OBJECT> tag.

HTML 4.0 standard supports the <OBJECT> tag, and this is expected to be the standard. If you want your code to be supported in the future, it is recommended you use the <OBJECT> tag before the <EMBED> tag.

Syntax

<ION_OBJECT
FILE="*filename*"
MIME="*mime-type*"
[DEBUG={"TRUE" | "FALSE"}]
[DELETE_FILE={"TRUE" | "FALSE"}]
[HEIGHT="*height*"]
[OBJTYPE={"EMBED | OBJECT"}]
[SERVER="*server name*"]
[WIDTH="*width*"]
[*Any attribute of the HTML <EMBED> or <OBJECT> tags*]>

[ION_PARAM tags]

```
        <IDL>
            IDL code used to write a file named filename
        </IDL>
</ION_OBJECT>
```

Attributes

In addition to the attributes listed below, it is usually necessary when using the HTML <OBJECT> or <EMBED> tag to specify an attribute that tells the plug-in where to retrieve the data that will be displayed by the plug-in. For the <OBJECT> tag, the DATA attribute is used for this purpose. For the <EMBED> tag, the SRC attribute is used. Some plug-ins ignore the DATA or SRC attribute and instead require a <PARAM> tag to provide this information. The method used to provide this information to the plug-in depends on the plug-in that is used to render the data. The $ION.IDLURL system variable defines the URL used to access the ION Script image engine, ion-i.exe. Set either the DATA attribute or SRC attribute to the value of the $ION.IDLURL system variable. See “Examples” on page 176 for examples that illustrate the use of the $ION.IDLURL system variable.

DEBUG

Setting this attribute to TRUE causes ION Script to generate a text file that can be used to help debug your ION Script application. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL. The file is given the name specified by the “Log Filename” setting on the “Debug” tab of the ION Script Configuration utility (Windows) or by the “ION Debug Filename” in the “Debug” section of the `.ionsrc` file (UNIX). For example, if the setting is “ion-out%d.txt”, then a file called `ion-out<PID>.txt` is created, where `<PID>` is the unique process ID for the instance of `ion-i` that generated the output. The file is written to the directory specified by the “Log Location” setting on the “Debug” tab of the ION Script Configuration utility (Windows) or by the “ION Debug Location” in the “Debug” section of the `.ionsrc` file (UNIX). In order for this to work, the directory specified by the “ION Debug Location” must be writable. See “Debug” on page 24 for more information. The default value for DEBUG is FALSE.

DELETE_FILE

Specifies whether to delete the file created in the <IDL> block. If TRUE, this file is deleted after it is streamed to the browser. If FALSE, the file is not deleted. The default value is TRUE.

HEIGHT

Specifies the height of the object. Although this attribute is optional, it is usually required by a plug-in.

FILE

Set this attribute to the name of the file written by IDL in the <IDL>...</IDL> block. Specify only the filename, not the path. The application assumes that this file is written to the directory specified by the $ION.temp system variable (see “temp” on page 58). The IDL code in the <IDL>...</IDL> block must create a file with the name specified by this attribute, and must write it to the path specified by the $ION.temp system variable.

MIME

Specifies the MIME type of the object.

OBJTYPE

Specifies whether to create the HTML <OBJECT> tag or <EMBED> tag. The default value is OBJECT.

SERVER

The full path to the image engine, `ion-i`. This must be a valid, absolute URL using the HTTP protocol. This attribute, which is also an attribute of <ION_DATA_OUT> and <ION_IMAGE>, can be used for load control by specifying different servers for different tasks (provided you have a copy of `ion-i` on each server, and that you have the proper license). Because the streaming of some types of data can be a resource-intensive process, it may be beneficial to performance in some cases to use a separate server to stream data inserted with the ION_OBJECT tag. If this attribute is not specified, ION Script will use the server specified during installation. The default value for this attribute can be changed using the configuration utility. This attribute can contain a variable.

WIDTH

Specifies the width of the object. Although this attribute is optional, it is usually required by a plug-in.

HTML Mapping

The ION_OBJECT tag is converted to either the HTML <OBJECT> tag (if the OBJTYPE attribute is set to OBJECT or if OBJTYPE is not specified), or the HTML <EMBED> tag (if the OBJTYPE attribute is set to EMBED).

Examples

This example illustrates the fact that the same MIME type is handled differently for Internet Explorer than for Netscape. When using the ION_OBJECT tag, it is usually necessary to code a different ION Script page for each browser that your application supports. In this example, `wav_ns.ion` is the Netscape version, which uses the Apple Quicktime plug in, and `wav_ie.ion` is the Internet Explorer version, which uses the Microsoft ActiveMovie plug- in.

wav_ns.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="file" TYPE="STR" VALUE="'test.wav'"/>
      <VARIABLE_DECL NAME="fullPath" TYPE="STR"
         VALUE="$ION.Temp+$file"/>
      <VARIABLE_DECL NAME="HTMLPath" TYPE="STR"
         VALUE="'http://'+$BROWSER.SERVER_NAME+'/'+$file" />
</VARIABLES>
</ION_HEADER>

<ION_BODY>
<ION_DATA_OUT>
   <IDL>
      x=findgen(5000)
      y=sin(x/5)*6000
      wave=fix(y)
      print,'$fullPath'
      write_wav,'$fullPath',y,10000

      print,'finished writing'
   </IDL>
</ION_DATA_OUT>
<ION_OBJECT HEIGHT="80" WIDTH="2500" FILE="$file" MIME="audio/wav"
   DATA="$ION.IDLURL">
   Wav file test
</ION_OBJECT>
</ION_BODY>
</ION_SCRIPT>
```

wav_ie.ion:

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="file" TYPE="STR" VALUE="'test.wav'"/>
      <VARIABLE_DECL NAME="fullPath" TYPE="STR"
         VALUE="$ION.temp+'\'+$file" />
</VARIABLES>
</ION_HEADER>

<ION_BODY>
<ION_DATA_OUT>
   <IDL>
      x=findgen(5000)
      y=sin(x/5)*6000
      wave=fix(y)
      print,'$fullPath'
      write_wav,'$fullPath',y,10000

      print,'finished writing'
   </IDL>
</ION_DATA_OUT>
<br>
<ION_OBJECT DEBUG="true" FILE="$file" MIME="audio/wav"
   ID="snd1" CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A">
      <ION_PARAM NAME="FileName" VALUE="$ION.IDLURL" />
</ION_OBJECT>
</ION_BODY>
</ION_SCRIPT>
```

ION_PARAM

The ION_PARAM tag is used to define object parameters inside the <ION_OBJECT>...</ION_OBJECT> tag pair. This tag is identical to the HTML <PARAM> tag, except that this tag is first processed by the ION Script parser. This allows you to use ION Script variables to define <PARAM> tags.

Syntax

<ION_OBJECT>

<ION_PARAM
[<*PARAM*> *attributes* (Can contain *$variable*)] />

</ION_OBJECT>

Attributes

Any attribute accepted by the HTML PARAM tag can be used with the ION_PARAM tag.

HTML Mapping

The ION_PARAM tag is converted to the HTML <PARAM> tag.

ION_RADIO

An ION_RADIO button is similar to an HTML radio button created with <INPUT TYPE="RADIO">. ION_RADIO allows you to maintain the selection state of a radio button group when reloading the page.

Syntax

<ION_RADIO
NAME="*group name*"
VALUE="*value*"
[DEFAULT]
[<*INPUT TYPE="RADIO"*> *attributes*] />

Attributes

DEFAULT

Include this attribute to cause this radio button to be the button that is initially selected when the page loads for the first time. Only one button in a group can include this attribute.

NAME

The name of the radio button group. Each radio button in a group uses the same NAME. This name is used to refer to the value of the selected radio button using the format $Form.*NAME*. This attribute can contain a variable. The NAME you specify for form fields cannot begin with ION_ _.

VALUE

The value assigned to the radio button, which can be accessed using $Form.*NAME*. If this value is numeric, and you want to use it in an expression, you must declare a new INT or DOUBLE variable, assign it the value $Form.*NAME*, and use the new variable in the expression instead of $Form.*NAME*. See “Using $Form Variables in Numeric Expressions” on page 54. This attribute can contain a variable.

Note

All attributes of the HTML <INPUT TYPE="RADIO"> are also available to ION_RADIO. See “INPUT” on page 129.

HTML Mapping

The ION_RADIO tag is converted to the HTML <INPUT TYPE="radio"> tag. The NAME and VALUE attributes of the ION_RADIO tag become the NAME and VALUE attributes of the <INPUT> tag. If DEFAULT is specified, it becomes the CHECKED attribute of the <INPUT TYPE="radio"> tag. All attributes of the HTML <INPUT> tag are also available to ION_RADIO.

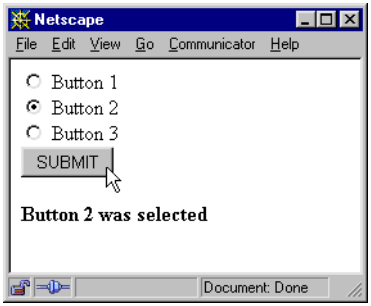
Example

In this example, we create a group of radio buttons, and a button with an EVENT that reloads the page. When the user clicks the button, the name of the selected button is written to the page. Note the following in this example:

- Each radio button in a group is given the same NAME attribute.
- When the page loads for the first time, the radio button given the DEFAULT attribute is initially selected.
- This example could also be created using radio buttons created with the INPUT tag instead of ION_RADIO, but when the page reloads, the state of the buttons would be lost.

```
<ION_SCRIPT>
<ION_HEADER>
   <EVENTS>
      <EVENT_DECL NAME="RELOAD" ACTION="ion://radio.ion" />
   </EVENTS>
</ION_HEADER>
<ION_BODY>
  <ION_FORM>
    <ION_RADIO NAME="Group1" VALUE="1"/> Button 1<BR>
    <ION_RADIO NAME="Group1" VALUE="2" DEFAULT/> Button 2<BR>
    <ION_RADIO NAME="Group1" VALUE="3"/> Button 3<BR>
    <ION_BUTTON EVENT="RELOAD" LABEL="SUBMIT" TYPE="BUTTON"/><BR>
  </ION_FORM>
  <ION_IF EXPR="NOT ($Form.Group1 ISTYPE 'UNDEF')">
    <B>Button <ION_VARIABLE NAME="$Form.Group1"/> was selected</B>
  </ION_IF>
</ION_BODY>
</ION_SCRIPT>
```

The following figure shows the result of this example:



ION_SCRIPT

The ION_SCRIPT tag pair is the container that holds all other ION Script tags, similar to the way the <HTML> tag contains all HTML tags. All content, whether it is ION Script or HTML, must reside inside the ION_SCRIPT tag pair.

Syntax

```
<ION_SCRIPT
        [SYNTAX = {"OFF" | "COMMENT" | "PRE" }]
        [<HTML> tag attributes] >
                ION Script tags
                [HTML tags]
   </ION_SCRIPT>
```

Attributes

SYNTAX

This attribute defines how expression parser error messages are reported for a given ION Script page. Valid values include:

- OFF — Error messages are not reported.
- COMMENT — Error messages are included in the returned HTML page as part of an HTML comment. This makes the error messages accessible but invisible to the client. To view the error messages, you must view the source of the HTML page sent to the browser by selecting **View → Page Source** (Netscape) or **View → Source** (Internet Explorer).
- PRE — Error messages are included in the returned HTML page as <PRE> text, and will appear at the bottom of the returned HTML page. This is useful during development of your ION Script applications.

The default method for reporting error messages is specified in the “Syntax Logging” section on the “Debug” tab of the ION Script Configuration utility (Windows), or in the “Debug” section of the `.ionsrc` file (UNIX). See “Syntax Logging” on page 25 for details. The SYNTAX attribute can be used to override the default for an individual ION Script page.

HTML Mapping

The ION_SCRIPT tag is converted to the <HTML> tag. All attributes of the <HTML> tag can be used in the <ION_SCRIPT> tag.

ION_VARIABLE

The ION_VARIABLE tag is used to insert the value of an ION Script variable into the processed document.

Syntax

<ION_VARIABLE NAME="$*variable*" [FORMAT="*format string*"] />

Attributes

NAME

The name of the variable. Note that when referencing an ION Script variable, the variable name must be prefixed with the $ symbol.

FORMAT

Specifies the C-style printf() formatting string. The syntax for the FORMAT attribute is as follows:

FORMAT="%[*flags*][*width*][.*precision*][*type length*] *specifier*"

where

flags is one or more of the following characters:

Flag	Description
space	Precede the result with a space
-	Left-justify the result within the field if width is specified
+	Precede the result with a sign (+ or -). The default is to display only a minus sign.
#	• If specifier is o, precede result with “0” • If specifier is x, precede result with “0x” • If specifier is X, precede result with “0X” • If specifier is e, E, f, g, G, the decimal point is always present in the result, even if no digits follow the decimal • If specifier is g, G, trailing zeros will display

Flag	Description
0	Pad the result with leading zeros (if the specified width is greater than the width of the result).

width is an integer specifying the minimum field width. If the result has fewer characters than the field width, it is padded on the left (or right, if the - flag has been specified). The padding is done with spaces unless the first character of *width* is a zero, in which case the padding is done with zeros.

precision is an integer preceded by a decimal point specifying the minimum number of digits to appear for the d, o, or x format specifiers, or the maximum number of characters to be printed for the s format specifier. It is also the precise number of digits to display after the decimal (including trailing zeros) for the e, E, f, g, or G specifiers.

type length is one of the following:

Type Length	Description
h	For d, i, o, u, x or X specifiers: value is treated as a short or unsigned short.
l (lower case L)	The value is treated as a long or unsigned long
L	The value is treated as a long double

specifier is one of the following:

ION Variable Type	Valid Specifiers
BOOL, STRING	%s
INT	%d, %i, %o, %u, %x, %X
DOUBLE	%e, %E, %f, %g, %G

Note

For more information on using format specifiers, refer to a C programming manual.

HTML Mapping

The ION_VARIABLE tag is not converted to HTML, and therefore does not accept any other attributes. The value of the specified variable is inserted directly into the HTML stream.

Example

In the following example, we declare a variable, and then insert the value of the variable into the page. Note that in the variable declaration, we use only the variable name without the $ symbol, but when referencing the variable in the ION_VARIABLE tag, we precede the variable name with the $ symbol:

```
<ION_SCRIPT>
<ION_HEADER>
   <VARIABLES>
      <VARIABLE_DECL NAME="foo" TYPE="INT" VALUE="5"/>
   </VARIABLES>
</ION_HEADER>
<ION_BODY>
   The value of foo is <ION_VARIABLE NAME="$foo"/>
</ION_BODY>
</ION_SCRIPT>
```

The following text is written to the Web page:

> The value of foo is 5

VARIABLE_DECL

The VARIABLE_DECL tag is used to declare a new variable in an ION Script application. The value for a variable can come from another variable, a numeric value, or an expression.

If the value you assign to a variable is an undefined variable or any expression that cannot be evaluated, the resulting value of the variable depends on the TYPE it was assigned. If TYPE is "INT" or "DOUBLE," the result is 0. If TYPE is "BOOL," the result is false. If TYPE is "STR," the result is a null string.

Note

The VARIABLE_DECL tag must reside between the <VARIABLES> and </VARIABLES> tags.

Syntax

```
<VARIABLE_DECL
        NAME="name"
        TYPE={"BOOL" | "INT" | "DOUBLE" | "STR"}
        VALUE="expression"
        [PERSIST={"TRUE" | "FALSE"}] />
```

Attributes

NAME

Defines the name of the variable. Do *not* precede the variable name with the $ symbol. This attribute is required. See Chapter 3, “Variables, Expressions & Operators” for information on variable names.

PERSIST

Set this attribute to TRUE to make the variable persistent. Persistent variables are used to maintain application state from one ION Script page to another. Once a variable is declared with PERSIST set to TRUE, it is available to all ION Script pages that are called without having to re-declare the variable in each page. If PERSIST is set to FALSE, the variable can only be referenced in the page in which it is declared.

TYPE

Defines the variable's data type. This attribute is required. The following table lists the allowable values for the TYPE attribute:

Attribute Value	Data Type
BOOL	Boolean
INT	Integer
DOUBLE	Double precision
STR	String

See "Variables" on page 44 for more on variable types.

VALUE

Defines the initial value of the variable. This attribute is required, and can contain a variable. If the type of data specified for VALUE differs from the specified TYPE, the value will be changed to the type specified by the TYPE attribute. See "Variable Types" on page 44 for a list of the cases in which the value is converted.

When setting the VALUE attribute to a string literal, the string should be enclosed in single quotation marks, all inside the double quotation marks that delimit the attribute value. For example, the following code shows the correct way to assign a string literal to the VALUE attribute:

```
<VARIABLE_DECL NAME="animal" TYPE="STR" VALUE="'dog'"/>
```

If you are assigning the VALUE attribute a string variable as opposed to a string literal, do not enclose the variable in single quotation marks. For example, if the variable animal has already been defined as a string, you would assign the value of the animal variable to a new variable as follows:

```
<VARIABLE_DECL NAME="pet" TYPE="STR" VALUE="$animal"/>
```

Warning

Assigning `$Browser.REMOTE_USER` as a string variable to the VALUE attribute can be dangerous because it is frequently an undefined system variable. When VALUE is undefined, IDL reports an error. However, it is possible that code that uses `$Browser.REMOTE_USER` could be implemented and tested without error on one system and fail if ported to another system.

HTML Mapping

The VARIABLE_DECL tag is not converted to HTML, and therefore does not accept any other attributes.

VARIABLES

The VARIABLES tag pair delimits a block of variable declarations in the header section of an ION Script document. The <VARIABLES>...</VARIABLES> block can contain one or more VARIABLE_DECL tags.

Note

The VARIABLES tag pair must reside between the <ION_HEADER> and </ION_HEADER> tags.

Syntax

<VARIABLES>

VARIABLE_DECL tags

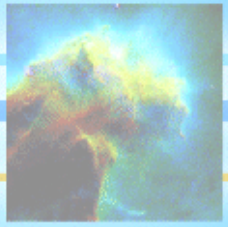
</VARIABLES>

Attributes

None

HTML Mapping

The VARIABLES tag is not converted to HTML, and therefore does not accept any other attributes.



Chapter 6:
Troubleshooting

If your ION Script application is not working as expected, start by checking the simple things first. Following is a list of some of the more common mistakes that may be causing problems with your application:

Web Server Problems

Before checking problems related to ION Script and IDL, make sure you are able to communicate with your Web server. For example, if you are using the Apache Web Server, you can enter a URL such as `http://localhost` or `http://hostname`, where *`hostname`* is the hostname or Windows machine name of your computer. If your Web server is properly configured, you should receive a page that says "If you can see this, it means that the installation of the Apache web server software on this system was successful." Consult your Web server documentation to find out how to test your Web server.

Also, make sure you are able to run CGI programs on your Web server.

If You Are Using Microsoft IIS

Using Microsoft IIS, the `.exe` extension must be included in URLs that point to executables. You will need to use `ion-p.exe` instead of `ion-p` in any URLs in your HTML and ION Script pages, and in the location field of your browser when requesting a URL.

"Connection Failed" Errors

If you get the message "ION Script: Error: Connection Failed" when you attempt to load an ION Script page, check the following:

- Does the page you are trying to load exist?
- If you are using Windows Notepad as your text editor, make sure `.txt` was not appended to the end of your `.ion` file. When you save a file in Notepad, such as `myfile.ion`, Notepad may automatically append `.txt` to the end, as in `myfile.ion.txt`. To avoid this, enclose the file in double quotation marks in the Save As dialog before saving the file.
- Does your URL point to the correct location of your `cgi-bin` directory?
- Does your "ION Search Path" include the directory containing the page you are trying to load? The "ION Search Path" can be set on the "Files" tab of the ION Script Configuration utility (Windows) or in the "Files" section of the `.ionsrc` file (UNIX). Remember that you cannot use the "+" symbol in the ION Search Path to indicate that all subdirectories of the specified directory should be searched. Therefore, each directory you want to be searched must be explicitly specified. See "ION Search Path" on page 20 for more information.
- Do both the page you are trying to load and the directory containing the file have read permission?

- If running ION Script on a UNIX Web server, does the case of the filename in the URL match the case of the actual filename?
- If you are able to run ION Script applications directly from your Web server, but external users are not able to run ION Script applications, check the Image Server URL and ION Script Parser URL values on the URLs tab of the ION Script Configuration utility (Windows), or in the URLs section of the `.ionsrc` file (UNIX). You must specify your Web server's fully qualified domain name and path to the CGI executables directory for your Web server. For example, the "Image Server URL" should look something like this:

  ```
  http://myhost.mydomain.com/cgi-bin/ion-i.exe
  ```

  If, during the installation process, you left the field blank on the "Web Server Configuration" dialog in which you specify your Web server's fully qualified domain name, the installer will use your Windows machine name, or your hostname on UNIX.
- Do you have an event named "page"? This is an illegal name for an event.

"Not Found" Errors

If you get an HTTP error informing you that the file or script you requested cannot be found, such as "HTTP Error 404," but the URL you are using appears to be correct, check the following:

- The required executables may not have been copied to your Web server's cgi-bin directory. Make sure that the files `ion-i.exe` and `ion-p.exe` (Windows) or `ion-i`, `ion-I`, `ion-p`, and `ion-P` (UNIX) are in your Web server's cgi-bin directory. If not, copy them from the `/ION61/ion_script/cgi-bin` subdirectory of your ION Script installation to your Web server's cgi-bin directory. On UNIX, make sure that the files `ion-i`, `ion-I`, `ion-p`, and `ion-P` have execute permission.

  Note that these executables would not be copied to the cgi-bin directory if, during the installation process, you left the field blank on the "Web Server Configuration" dialog in which you specify the CGI executables directory.
- If you are using Microsoft IIS, the `.exe` extension must be included in URLs that point to executables. You will need to use `ion-p.exe` instead of `ion-p` in any URLs in your HTML and ION Script pages, and in the location field of your browser when requesting a URL.

“No Event or Page” Errors

- If you get “Error: No event or page” when you use the Enter key to submit an ION_FORM containing a single text field, you need to associate an event with the text field by defining an event with the same name as the NAME attribute of the text field. For an example, see “Forms Containing a Single Text Field” on page 153.

Licensing Errors

If you get a page with an error message such as “License Error: RSI license checkout failed”, ION Script may not be properly licensed. View the licensing instructions in the *Installing and Licensing IDL* manual and make sure you have properly licensed ION.

Note

Since Web Servers do not read any system environment variables, you cannot use the LM_LICENSE_FILE environment variable to point to where you have located your license file. When licensing ION Script, you must place your license file in the default location *`RSI-DIR`*`\license\license.dat` where *`RSI-DIR`* is the name of the main installation directory you selected to install ION Java. On UNIX, if you have a network license that you wish your ION installation to use, you can copy the network license file to the default location *`RSI-DIR`*`\license\license.dat` on the machine on which you are running ION.

ION Script Syntax Errors

- Make sure all single tags use the proper syntax. Single tags are closed with />. Make sure you didn’t mistakenly close a single tag using only >, or use tag-pair syntax for a single tag. ION Script will return an error message to inform you of a mismatched tag pair. For example, consider the following code:

```
<ION_BODY>
   <ION_VARIABLE NAME="$A">
</ION_BODY>
```

  ION Script will return the following error:

  Error: Mismatched tag pair: <ION_VARIABLE> ... </ION_BODY>

  This error message indicates that you opened a tag called <ION_VARIABLE>, and ION Script found the next closing tag </ION_BODY>. This should alert you to the incorrect syntax for ION_VARIABLE (i.e., the proper way to close this tag is with /> rather than just >).

- Make sure all tag pairs have both the opening and closing tag. For example, if you have an <ION_HEADER> tag, you must also have an </ION_HEADER> tag.
- Check the case and spelling of your tags, attribute names, and system variables. Make sure you've used the exact case shown in the Tag Reference.
- Make sure you have specified all the required attributes of a tag. In the syntax listings, attributes are required unless they are surrounded by square brackets.
- Make sure all attribute values have been enclosed in double quotation marks. Remember to include both opening and closing quotation marks.
- If you get the error "Illegal character ["] during parsing", make sure you have not included a carriage return between the double quotation marks that delimit the value of an ION Script attribute. For example, the following ION Script code is *not* allowed:

```
<ION_EVALUATE EXPR="$X +
   $Y"/>
```

IDL Errors

- Set the DEBUG attribute of the ION_DATA_OUT and ION_IMAGE tags to TRUE to cause a debugging file to be generated in the directory specified by the "ION Debug Location" setting on the "Debug" tab of the ION Script Configuration Utility (Windows) or in the "Debug" section of the `.ionsrc` file (UNIX). See "DEBUG" on page 142. This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL.
- Make sure your <IDL> blocks do not use IDL commands that cannot be entered at the command line such as a FOR loop or BEGIN statement.
- Make sure you are not using any other IDL commands that are not allowed in ION Script. See "IDL Tag Limitations" on page 124.
- If your <IDL> block contains a number (or an ION Script variable that begins with a number) enclosed in quotation marks, make sure you enclose the number or variable in single rather than double quotation marks. In IDL, a double quotation mark followed by a number denotes an octal constant. For example, the following code would cause a syntax error in IDL:

  ```
  print, "1 Elm Street"
  ```

  Instead, you would need to use

  ```
  print, '1 Elm Street'
  ```

  See "Single vs. Double Quotation Marks" on page 126 for more information.

- If you get an “Attempt to call undefined procedure/function” error, make sure the `.pro` or `.sav` file containing the procedure or function you are calling is located in a directory specified in the IDL Search Path. On UNIX, also make sure all directories in the IDL Search Path have the proper read permissions. Set the IDL Search Path on the “Files tab” of the ION Script Configuration utility (Windows) or in the “Files” section of the `.ionsrc` file (UNIX).

Variables

- Check user-defined variables to make sure that each variable is used consistently with regard to case and spelling throughout the document. For example, make sure you didn’t declare a variable as $A, and later assign a value to $a.
- When checking the value of a boolean variable, such as in the statement $A EQ TRUE, make sure you use either all uppercase or all lowercase letters for the boolean value. In other words, use only TRUE, true, FALSE, or false. Do not use True or False.
- Make sure you didn’t mistakenly declare a variable with a value that is of a different type than the specified TYPE. For example, if you assign a value of 4.99 to a variable declared as an INT, the value becomes 4.
- When using a form variable in a numeric expression, make sure you declare a variable with the same value as the form variable and assign a numeric TYPE. See “Using $Form Variables in Numeric Expressions” on page 54.
- If a value is being passed to an ION Script page in a URL, make sure you refer to this value as a $Form variable in the page to which the value is being passed. For example, assume you want to pass the parameter DATA with a value of 34 to the page `mypage.ion` with the following URL:

  ```
  http://myserver/cgi-bin/ion-p?page=mypage.ion&DATA=34
  ```

  In `mypage.ion`, in order to access the DATA parameter, you must refer to it as $Form.DATA, not $DATA.

Images Not Displaying

If your ION Script page contains an image from an <ION_IMAGE> tag, and a box is being displayed in place of the image you expect, check the following:

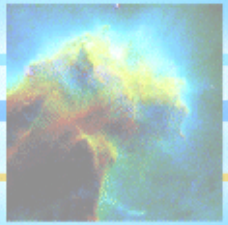
- Make sure there are no IDL errors occurring in your <IDL> block. To do this, set the DEBUG="TRUE" in the ION_IMAGE tag to cause a debugging file to be generated in the directory specified by the ION Debug Location setting on the Debug tab of the ION Script Configuration utility (Windows) or in the

Debug section of the `.ionsrc` file (UNIX). This debugging file contains the exact code sent to IDL, as well as any errors reported by IDL.

- If your image is created from a `.pro` file, make sure the `.pro` file is in the IDL Search Path. The IDL Search Path can be set on the Files tab of the ION Script Configuration utility (Windows) or in the Files section of the `.ionsrc` file (UNIX).
- Make sure you have specified the correct TYPE attribute for the ION_IMAGE tag. If the code in your IDL block uses IDL Object Graphics, the TYPE attribute must be set to OBJECT. If the code in your IDL block uses IDL Direct Graphics, the TYPE attribute must be set to DIRECT.

Frames

- When using frames, make sure the page containing the <FRAMESET> does not contain an HTML <BODY> tag or an ION Script <ION_BODY> tag.



Index

Symbols

" (double quotation mark), 116, 195
$ symbol
  IDL line continuation, 127
  printing in IDL, 127
  variables in IDL, 127
$Browser system variable, 50
$Document system variable, 51
$Form system variable, 53
$Form variables, 196
$FormURL system variable, 54
$ION.IDLURL system variable, 58
$ION.temp system variable, 58
$ION.uniqueID system variable, 59
$ION.version system variable, 59
$Mouse system variable, 59, 107
( ) (to group expressions), 67
* operator, 64
+ operator (string concatenation), 64
+, - operators, 64
/ operator, 64
<!-- --> (comment tag), 119
[ ] (square brackets), 115
\ (backslash character), 128
^ operator, 64
_blank, 153, 162, 170
_parent, 153, 162, 170
_self, 153, 162, 170
_top, 153, 162, 170
{ } (braces), 116
’ (single quotation mark), 116

Numerics

24-bit images, 92
8-bit character sets, 94
8-bit images, 92

A

ACTION attribute
  EVENT_DECL, 122
  FORM tag (HTML), 36
addition operator, 64
ALINK attribute, 134
AND operator, 66
APPLICATION tag, 156
applications, performance, 92
ASCII character sets, 94
ASTEXT attribute
  ION_DATA_OUT, 141
  ION_INCLUDE, 165
attributes, using variables in, 98, 117
AUTHOR tag, 156

B

BACKGROUND attribute, 134
backslash character, \, 128
bandwidth, 92
BGCOLOR attribute, 134
BODY tag, 197
BOOL type, 44, 187
boolean expressions, 62
BORDER attribute
  ION_BUTTON, 135
  ION_IMAGE, 160
braces, 116
brackets
  curly, 116
  square, 115
Browser system variable, 50
browsers
  determining type of, 50
  support of frames, 89
buttons
  INPUT, 129
  ION_BUTTON, 135
  radio, 179

C

carriage returns, 54
case sensitivity, 47, 116, 195
CGI, 36
character set, 94
characters
  / (in URLs), 72, 73
  \ (backslash), 128
checkboxes
  INPUT, 129
  ION_CHECKBOX, 138
colors, specifying, 133
comment tag, 119
comments
  HTML, 119
  IDL, 127
comparison operators, 65
compression, image, 92
CONTAINS operator, 64
COPYRIGHT tag, 157
curly brackets, 116
cursor, determining where clicked, 59

D

data, types, 187
DATE tag, 157
DEBUG attribute
  ION_DATA_OUT, 142
  ION_IMAGE, 160
  ION_OBJECT, 174

debugging
DEBUG attribute, 142, 160
single tags, 114
troubleshooting, 191
using the comment tag, 119
declaring variables, 44, 98, 186, 189
DEFAULT attribute, ION_RADIO, 179
DELETE_FILE attribute, ION_OBJECT, 174
direct graphics, 91, 159
DISPLAY attribute, ION_EVALUATE, 149
division operator, 64
Document system variable, 51
double quotation marks, 126
DOUBLE type, 44, 187

E

entity names, 94
entity references, 94
EQ operator, 65
error messages, 191
error reporting, 104
errors
licensing, 191
Not Found, 191
syntax, 25, 182, 191
troubleshooting, 191
undefined procedure/function, 196
EVENT attribute
ION_BUTTON, 136
ION_IMAGE, 161
ION_LINK, 170
EVENT_DECL tag, 122
events
declaring, 122, 123
determining name of, 52
EVENTS tag, 123
examples, ION Script, 41
exponentiation operator, 64
EXPR attribute
ION_EVALUATE, 54, **149**
EXPR attribute (*continued*)
ION_IF, 54, **158**
expression types, 61
expressions
boolean, 62
conditionally evaluating, 158
evaluating, 149
examples, 61
grouping, 67
numeric, 62
string, 63
values in, 61

F

FILE attribute, ION_OBJECT, 175
file formats, 92
files
including, 165
location of, 38, 84
Form system variable, 53
form variables, 130, 170, 196
FORMAT attribute, ION_EVALUATE, 150, 183
formatting
IDL output, 143
included files, 166
forms
creating, 100
fields, 152
HTML vs. ION Script, 74
ION Script, 152
validating, 104
FormURL system variable, 54
frames
accommodating all browsers, 89
fixed, 82
floating, 87
locating source files, 84
targeting different, 153, 162, 170
troubleshooting, 197

FRAMESET tag, 197

G

GE operator, 65
GET method, 136, 143, 152
graphics
  direct vs. object, 91
  selecting image format, 92
grouping expressions, 67
GT operator, 65

H

header section, 156
HEIGHT attribute
  ION_IMAGE, 161
  ION_OBJECT, 175
hidden form fields, to pass name/value pairs, 36
htdocs directory, 38
HTML files
  including, 165
  loading ION Script pages from, 36
  location of, 38, 84
hypertext links, 170

I

IDL
  commenting, 127
  creating dynamic HTML with, 145
  creating images with ION Script, 159
  direct graphics, 91
  formatting output, 143
  inserting output into ION Script, 141
  line continuation, $, 127
  object graphics, 91
  programs, large, 143
  using in ION Script, 124
IDL tag, **124**
IDLgrBuffer object, 91
IDLURL, $ION system variable, 58
IF statement, 158
image formats, 92
images
  ALT text, 161
  borders, 160
  creating, 159
  events from clicking, 107, 161
  file format, 161
  labels, 161
  not displaying, 191
  size of, 92, 161
IMG_TYPE attribute
  choosing value for, 92
  ION_IMAGE tag, 161
including files, 106, 165
INPUT tag, 129
INT type, 44, 187
international characters, 94
ION Script
  applications, 32
  architecture, 33
  examples, 41
  overview, 28
  pages, 28
  requesting a page, 36
ION Script files, location of, 38, 84
ION_BODY tag, 197
ION_BUTTON tag, 135
ION_CHECKBOX tag, 138
ION_DATA_OUT tag, 108, **141**
ION_ELSE, 158
ION_ELSEIF, 158
ION_EVALUATE tag, 44, 54, **149**
ION_FORM tag, 100, **152**
ION_HEADER tag, 156
ION_IF tag, 54, **158**
ION_IMAGE tag, 159
ION_INCLUDE tag, 106, **165**
ION_LINK tag, 170
ION_OBJECT tag, 173

ION_PARAM tag, 178
ION_RADIO tag, 179
ION_SCRIPT tag, 182
ION_SCRIPT_BUFFER, 91, 159
ION_SCRIPT_DEST, 91
ION_VARIABLE tag, 99, **183**
ISO 8859, 94
ISTYPE operator, 46
italics, *See* syntax

J

JPEG images, 92, 161

L

LABEL attribute, 136
  ION_IMAGE tag, 161
LASTUPDATE tag, 157
LE operator, 65
licensing, errors, 191
line continuation, IDL, 127
LINK attribute, 134
links
  ION Script, 170
  using VALUE attribute, 145
logical operators, 66
lossless compression, 92
lossy compression, 92
lower case, *See* case sensitivity
LT operator, 65

M

mathematical operators, 64
MAXLENGTH property, 152
messages, error, 191
metadata tags, 156
METHOD attribute
  ION_BUTTON, 136
  ION_DATA_OUT, 143
METHOD attribute (*continued*)
  ION_FORM, 152
MIME attribute, ION_OBJECT, 175
MOD operator, 64
Mouse system variable, 59, 107
multiplication operator, 64

N

NAME attribute
  EVENT_DECL, 122
  ION_CHECKBOX, 138
  ION_FORM, 153
  ION_LINK, 170
  ION_RADIO, 179
  ION_VARIABLE, 183
  VARIABLE_DECL, 186
name/value pairs, 36, 53, 86, 196
NE operator, 65
nesting quotation marks, 116
numeric expressions, 62

O

object graphics, 91, 159
OBJTYPE attribute, ION_OBJECT, 175
ONLOAD attribute, 134
ONRESET attribute, 153
ONSUBMIT attribute, 153
ONUNLOAD attribute, 134
operators
  comparison, 65
  logical, 66
  mathematical, 64
  precedence of, 67
  string, 64
OR operator, 66
origin, location of, 59
output formats, 92

P

parentheses, 67
performance, 92
PERSIST attribute
  using, 49
  VARIABLE_DECL tag, 186
persistence
  form variables, 54
  variable, 49, 186
PNG images, 92, 161
POST method, 136, 143, 152
PRE attribute
  ION_DATA_OUT, 143
  ION_INCLUDE, 166
precedence, operator, 67
process ID, 59

Q

query strings, 53
quotation marks, 116, 126, 195

R

radio buttons
  creating, 180
  determining selected, 180
  INPUT, 129
  ION_RADIO, 179
reserved words, 47
RGB triplets, 133

S

SERVER attribute
  ION_DATA_OUT, 143
  ION_IMAGE, 162
  ION_OBJECT, 175
single quotation marks, 126
SIZE property, 152
special characters
  converting to character entities, 165
  ISO 8859, 94
  URL encoding, 54
square brackets, 115
SRC attribute
  ION_BUTTON, 136
  ION_INCLUDE, 166
storing files, 38
STR type, 44, 187
strdecode.pro, 55
string expressions, 63, 187
string literals, 46
string operators, 64
strings
  decoding, 55
  defining in IDL, 126
strongly typed variables, 44
SUBMIT form field, 152
subtraction operator, 64
syntax, URLs, 70
SYNTAX attribute, 182
syntax conventions, 114
syntax errors, 191
system variables, 50
  $Browser, 50
  $Document, 51
  $Form, 53
  $FormURL, 54
  $ION.IDLURL, 58
  $ION.temp, 58
  $ION.uniqueID, 59
  $ION.version, 59
  $Mouse, 59
  Form, 100
  Mouse, 107

T

TABINDEX property, 152
tabs, 54

tags
  comment, 119
  mismatched, 191
TARGET attribute
  ION_FORM, 153
  ION_IMAGE, 162
  ION_LINK, 170
temp files, specifying location, 21
temp, $ION system variable, 58
temporary files, location, 58
TEXT attribute, 134
text files
  including, 165
  specifying extensions of, 21
TEXT form field, 152
TITLE tag, 157
troubleshooting, 191
TYPE attribute
  ION_BUTTON, 137
  ION_IMAGE, 162
  VARIABLE_DECL, 187

U

undefined procedure/function error, 196
undefined variables, checking for, 46
uniqueID, $ION system variable, 59
upper case, *See* case sensitivity
URL-encoding, 54
URLs
  file protocol, 72
  http protocol, 71
  ion protocol, 72
  passing data in, 53, 86, 196
  specifying, 70

V

validating form data, 104
VALUE attribute
  ION_CHECKBOX, 138
  ION_FORM, 152
  ION_RADIO, 179
  VARIABLE_DECL, 187
values in expressions, 61
variable substitution, 98, 117
VARIABLE_DECL tag, 186
variables, 44
  assigning string literals, 46
  assigning values to, 44, 187
  declaring, 44, 44, 98, 186, 189
  form, troubleshooting, 196
  form, using in expressions, 54
  in HTML attributes, 117
  in ION Script attributes, 98, 117
  inserting value of, 99, 183
  names, 47, 186
  persistence, 49, 186
  system, 50
  type conversion, 196
  type, determining, 46
  types, 44
  undefined, 46
VARIABLES tag, 189
version, $ION system variable, 59
version, browser, 50
VLINK attribute, 134

W

whitespace, preserving, 143, 166
WIDTH attribute, 162
  ION_OBJECT, 175
WINDEX keyword, 159
WINDOW procedure, 124
windows, creating IDL, 124
word-wrap, preventing, 143, 166

Z

z-buffer, 91